**VŠB – Technická univerzita Ostrava**
**Fakulta elektrotechniky a informatiky**
**Katedra telekomunikační techniky**

# Kolokované nasazení Asterisk PBX a Kamailio SIP Proxy ve funkci SBC

# Combined Implementation of Asterisk PBX and SIP Proxy Kamailio in the Role of the SBC

**2017** **Bc. Róbert Šimovič**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

# Zadání diplomové práce

Student: **Bc. Róbert Šimovič**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Kolokované nasazení Asterisk PBX a Kamailio SIP Proxy ve funkci SBC
Combined Implementation of Asterisk PBX and SIP Proxy Kamailio in the Role of the SBC.

Jazyk vypracování: čeština

Zásady pro vypracování:

Student si nastuduje problematiku implementace SIP signalizace a zpracování médií v dvou nejběžněji používaných SIP serverech - Asterisku a Kamailiu. Na základě získaných znalostí pak navrhne a implementuje systém plnící základní funkce SBC prostřednictvím spojení výhod obou zmíněných SIP serverů provozovaných kolokovaně na jednom fyzickém/virtuálním serveru. Student provede výkonnostní měření navrženého systému a na základě výsledků zhodnotí vhodnost a konkurenceschopnost navrženého řešení.

Cíle práce pak shrnují následující body:
1. Student si nastuduje a zhodnotí architektury a implementace SIP serverů Asterisk a Kamailio.
2. Student identifikuje klíčové funkce, postupy a systémy nutné pro kombinované nasazení SIP serverů Asterisk a Kamailio.
3. Student definuje minimální soubor funkcí SBC realizovatelných prostřednictvím Asterisk PBX a Kamailio SIP Proxy.
4. Student navrhne SBC využívající kombinace Asterisk PBX a Kamailio SIP proxy nasazené na jednom serveru a implementuje funkce definované v předchozím bodě.
5. Student zhodnotí vytvořené řešení z pohledu jeho výkonu a konkurenceschopnosti.

Seznam doporučené odborné literatury:

1. Bryant, R. Madsen, L. Asterisk: The Definitive Guide. ISBN 1449332420.
2. Goncalves, F. Building Telephony Systems with OpenSIPS 1.6. ISBN 1849510741.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Rozhon, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Ing. Miroslav Vozňák, Ph.D.
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

## Prehlásenie študenta

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave, dňa: *28-aprila-2017*

Samostí
.....
podpis študenta

# Poďakovanie

Rád by som poďakoval Ing. Janovi Rozhonovi, Ph.D., za odbornú pomoc a konzultáciu pri vytváraní tejto diplomovej práce.

# Abstrakt

Práca sa zaoberá vytvorením SBC pomocou spoločného nasadenia pobočkovej ústredne Asterisk a SIP proxy Kamailio.

Pri návrhu SBC bolo potrebné brať do úvahy spôsob implementácie oboch platforiem na spoločne zdieľanom virtuálnom serveri. Návrh je vystavaný na novších verziách platforiem a využíva tie výhody oboch platforiem, ktoré robia dané riešenie atraktívne. Kľúčové funkcie pre celý systém sú bezpečnosť a variabilita nastavenia SBC, ktoré je možné meniť podľa aktuálnych požiadaviek. V mojom realizovanom prostredí bolo potrebné navrhnúť súbor bezpečnostných funkcií, ktoré naplňovali funkciu SBC.

Celý systém bol vystavený výkonnostnému testovaniu, ktorý som vyhodnocoval pomocou grafov. Zásadou metodiky testovania je možnosť reprodukovateľnosti.



# Kľúčové slová

Kamailio, Asterisk, SBC, testovanie, SIPp, bezpečnosť, VoIP, HTABLE, PIKE, TOPOH, RATELIMIT, TLS

# Abstract

This thesis deals with the creation of SBC by the combined implementation of Asterisk PBX and SIP Proxy Kamailio.

When designing the SBC, it was necessary to take in consideration the implementation of both platforms on one share virtual server. The design is based on newer versions of mentioned platforms and uses the benefits of both platforms, which make the solution more attractive. The key functions of the whole system are security and variability of the SBC settings, which can be altered according to actual requirements. In my implemented environment, it was necessary to design a set of security features, which fulfilled the function of SBC.

The whole system has been subjected to performance testing, which I have evaluated by using graphs. The principle of testing methodology is a possibility of reproducibility.



# Key words

Kamailio, Asterisk, SBC, testing, SIPp, security, VoIP, HTABLE, PIKE, TOPOH, RATELIMIT, TLS

# Obsah

# Zoznam použitých skratiek

| **Skratka** | Anglický význam | Slovenský význam |
|---|---|---|
| **ACK** | Acknowledge | Potvrdenie prijatia |
| **B2BUA** | Back-to-Back User Agent | Varianta SIP Serveru |
| **CLI** | Commnad Line | Príkazový riadok |
| **CSV** | Commna Separated Values | Formát textového súboru |
| **DDoS** | Distributed Denial of Service | Distribuované odmietnutie služby |
| **DoS** | Denial of Service | Odmietnutie služby |
| **GPL** | General Public License | Všeobecná verejná licencia |
| **HW** | Hardware | Fyzické zariadenie |
| **IETF** | Internet Engineering Task Force | Komisia správy internetu |
| **IM** | Instant Messaging | Okamžité správy |
| **IoT** | Internet of Things | Internet vecí |
| **IP** | Internet Protocol | Internetový protokol |
| **IPv4** | Internet Protocol version 4 | Internetový protokol verzie 4 |
| **IPv6** | Internet Protocol version 6 | Internetový protokol verzie 6 |
| **LDAP** | Lightweight Directory Access Protocol | Protokol ľahkého prístupu k adresáru |
| **PBX** | Private Branch Exchange | Pobočková ústredňa |
| **RFC** | Request for Comments | Žiadosť o komentáre |
| **RTP** | Real-Time Transport Protocol | Protokol pre prenos v reálnom čase |
| **SBC** | Session Border Controller | Kontrola relácii na hranici siete |
| **SCTP** | Stream Control Transmission Protocol | Protokol riadenia prenosu toku |
| **SDP** | Session Description Protocol | Protokol popisu relácie |
| **SER** | SIP Express Router | SIP smerovač |
| **SIP** | Session Initiation Protocol | Protokol inicializácie relácie |
| **SNMP** | Simple Network Management Protocol | Protokol jednoduchého sieťového manažmentu |
| **SQL** | Structured Query Language | Štruktúrovaný vyhľadávací jazyk |
| **SSH** | Secured Shell | Zabezpečené vzdialené prihlásenie |
| **SW** | Software | Programové zariadenie |
| **TCP** | Transmission Control Protocol | Protokol riadenia prenosu |

| | | |
|---|---|---|
| **TLS** | Transport Layer Security | Protokol šifrovania prenosu |
| **UA** | User Agent | Koncový účastník |
| **UDP** | User Datagram Protocol | Používateľský datagramový protokol |
| **URI** | Universal Resource Identifier | Jednotný identifikátor prostriedku |
| **VoIP** | Voice over Internet Protocol | Hlas cez internetový protokol |
| **WebRTC** | Web Real-Time Communication | Prenos v reálnom čase na internete |
| **XML** | Extensible Markup Language | Rozšíriteľný značkovací jazyk |

# Zoznam ilustrácií

# Zoznam tabuliek

# Úvod

V dobe zvyšujúceho sa počtu kybernetických útokov je potrebné čoraz viac napredovať v možnostiach zabezpečenia hraničných bodov na sieti, kde prebieha výmena informácií medzi vnútornou a vonkajšou sieťou. Jedným, z takýchto výmen informácií, t.j. spojení je aj hlasová komunikácia, ktorej sa budem venovať v tejto práci. Jedná sa o veľmi citlivé dáta, ktoré si neželáme stratiť počas spojenia a taktiež ubezpečiť sa o ich zabezpečenom prenose. Mojou úlohou bude vytvorenie hraničného bodu na sieti vo forme čiernej skrinky, o ktorom potenciálny útočník nemôže vedieť akú má formu, t.j. architektúru a taktiež jej zabezpečovacie techniky, t.j. spôsoby obrany pred rôznymi útokmi ako je flooding, BFA a DDoS. Jednou z foriem zabezpečenej formy výmeny informácií bude aj TLS.

Virtualizácia je často odpoveďou na vytvorenie systému, ktorý je flexibilný a ľahko nahraditeľný pri strate, poškodení či znefunkčnení útočníkom. Preto aj v tejto práci bude využitá virtualizácia pomocou systému VMware. Vďaka tomu môžeme nahradiť poškodený systém pomocou vytvoreného obrazu celého systému v prípade negatívnych činiteľov, respektíve aspektov, ktoré znefunkčnia či odcudzia daný komunikačný systém.

Aby som sa mohol dopracovať k tak pokročilým témam, musím na úvod taktiež popísať aj základnú funkciu SIP signalizácie a rozobrať SIP správu. Predstavím pobočkovú ústredňu Asterisk a SIP Proxy Kamailio. Spomeniem nástroj na zhodnotenie prenosu informácií na sieti, t.j. program Wireshark a výkonnostný testovací program SIPp.

V ďalších sekciách prejdem na návrh SBC a implementácie SIP proxy Kamailio a pobočkovej ústredne Asterisk na spoločný server. Využitie nájde aj databázový systém MySQL a predstavím aj výhody spoločného spojenia SIP proxy Kamailio a pobočkovej ústredne Asterisk.

Požiadavky na bezpečnostné prvky budú letmo načrtnuté na záver teoretickej časti a ich implementáciu sa uvediem až v praktickej časti.

# 1 Úvod do technológie komunikácií

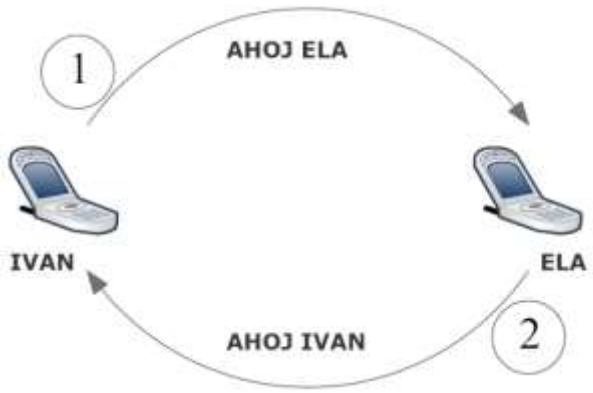
Ako aj na Slovensku tak aj vo svete nastal prudký rozvoj internetu a s ním spojených služieb. Či dátových, multimediálnych, hlasových, e-mailových a webových služieb. Vzrast nastal popularitou internetu, jeho možností ako komunikovať, vymieňať si e-maily, krátke správy, či komunikovať pomocou hlasu alebo videa. Taktiež internet sa stal vo vyspelých západných krajinách dostupnou službou pre kohokoľvek, bez ohľadu na vek, fyzický či psychický stav, farbu pleti či vierovyznania.

Akákoľvek komunikácia, ktorá nastane na internete, aby bola rozpoznateľná pre obidve strany a taktiež sprostredkovateľmi daného prenosu, musí byť definovaná podľa štandardu. Štandard určuje spôsob tvorenia správ, ich obsah a protokolovú stavbu správy. Bez dodržania štandardov by len prúdili toky správ, ktoré je ťažšie dekódovať, rozpoznať, aký majú cieľ, či od koho pochádzajú, čo je ich obsahom, či aký typ dekódovania obsahu treba použiť.

Neexistuje organizácia, ktorá by bola formálne zodpovedná za internet. Existuje, ale organizácia, ktorá hrá rolu tvorcu štandardov pre internet. Je ňou IETF (Internet Engineering Task Force), ktorá sa sformovala začiatkom roka 1986, čo je už 31 rokov neustáleho vývoja nových štandardov. Tvorí špecifikácie pre prenos a smerovanie informácií a dát na internete, technicky známych ako pakety. IETF organizácia vytvorila protokoly pre e-mail, rozpoznávanie a preklad adries.

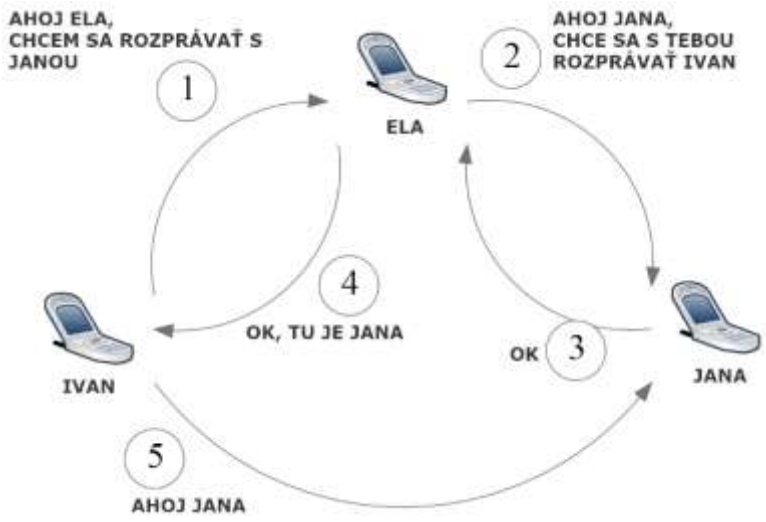
Ako som spomenul, organizácia sa dlhšiu dobu venuje aj tvorbe štandardov pre prenos hlasových služieb. Organizácia IETF vytvorila pod svojou záštitou viacero štandardov pre hlasovú IP telefónnu komunikáciu. Je ním protokoly RTP, ktorý slúži na výmenu hlasových a video dát, protokol SDP, ktorý slúži na vyjednanie spojenia a popis multimediálnych dát, ktoré budú využité pri komunikačnej službe. Jedným z veľkých pokusov, ktorý mal zabezpečiť akúkoľvek komunikáciu, nielen hlasovú službu, taktiež združiť signalizáciu a vyjednanie spojenia je protokol SIP, o ktorom budem písať neskôr.

Proces vytvorenia VoIP telefonátu pochádza z bežnej komunikácie. Existujú dva druhy správ, signalizačných správ, ktoré na seba nadväzujú. Sú nimi žiadosť a odpoveď. Názorný príklad jednouchej komunikácie, t.j. odpovede na žiadosť zobrazuje Ilustrácia 1. Takáto priama komunikácia sa používa najmä v systémoch, ktoré sa navzájom poznajú a môžu sa kontaktovať bez akéhokoľvek prostredníka, t.j. spojovacieho bodu. Také riešenie nie je časté a jeho využitie je vhodné len pre malý počet klientov.




*Ilustrácia 1 - Priama komunikácia medzi dvomi užívateľmi*

Takéto priame spojenie vo svete internetu či internej siete je nemožné. Vždy existuje spojovací bod, ktorý sprostredkuje výmenu informácií, t.j. žiadostí a odpovedí medzi koncovými účastníkmi, t.j. UA. Ilustrácia 2 zobrazuje takýto prepojovací bod medzi tromi užívateľmi.




*Ilustrácia 2 - Sprostredkovaná komunikácia medzi tromi užívateľmi*

Keďže vývoj a požiadavky na hlasový systém sa neustále zvyšujú, tak základná vlastnosť sprostredkovať komunikáciu bola dostatočná ešte v dobe keď siete boli malé a nie tak rozsiahle ako v dnešných dňoch. Zákazníci požadujú druh tzv. kvality služby (Quality of Service), rôzne štatistické prehľady vyťaženia systému a najmä ochranu pred útokmi od možných útokov z vonkajšej siete. Keďže korporácie, univerzity, stredné či malé firmy sú pripojené 24 hodín denne, 7 dní v týždni do internetu, je vhodné pre ochranu odcudzenia dát zabezpečiť ochranu daných hlasových služieb. Častokrát útočníci len chcú znefunkčniť daný systém, aby ho nebolo možné používať a požadujú za ukončenie svojich útokov finančnú odmenu. Jedným z celistvých riešení je aplikácia hlasovej služby vo forme SBC (Session Border Controller), ktorá zabezpečuje ochranu a sprostredkovanie výmeny SIP signalizácie v sieti. O možnostiach ako vytvoriť SBC budem písať neskôr v kapitole č. 2 a jeho samotné vlastnosti popíšem v podkapitole 2.2.

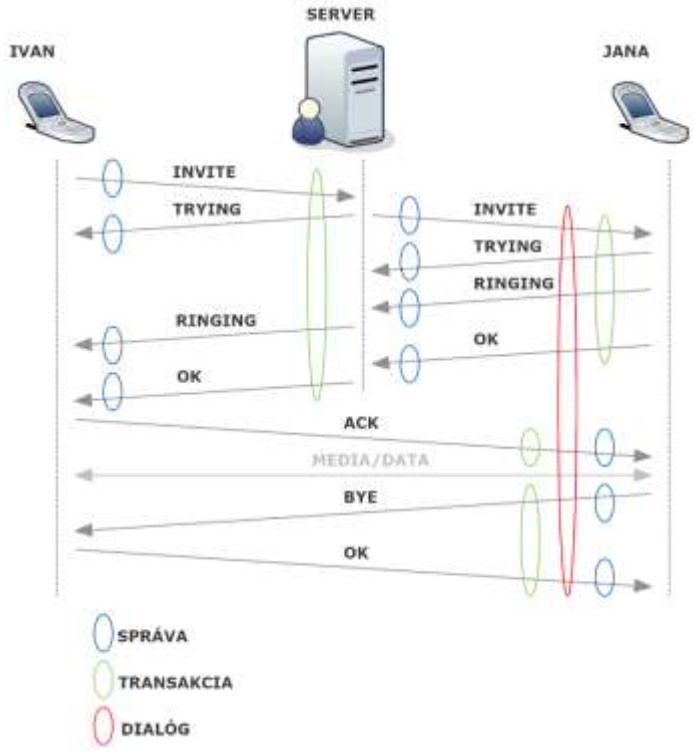
## 1.1. Krátky úvod do protokolu SIP

Protokol SIP (Session Initiation Protocol) je signalizačný protokol, ktorý pracuje na relačnej vrstve OSI modelu a jeho úlohou je vytváranie, úprava a ukončovanie spojení s jedným alebo viacerými účastníkmi. Tieto spojenia zahrňujú internetové telefonáty, multimediálnu distribúciu a multimediálne konferencie. Je vyvinutý organizáciou IETF (Internet Engineering Task Force) a vydaný pod RFC 3261 [1].

SIP žiadosti sa využívajú na vytvorenie spojení, ktoré nesú informácie o popise daného spojenia. To umožní účastníkom dohodnúť sa na súbore zhodných multimediálnych nastavení pri nasledovnom

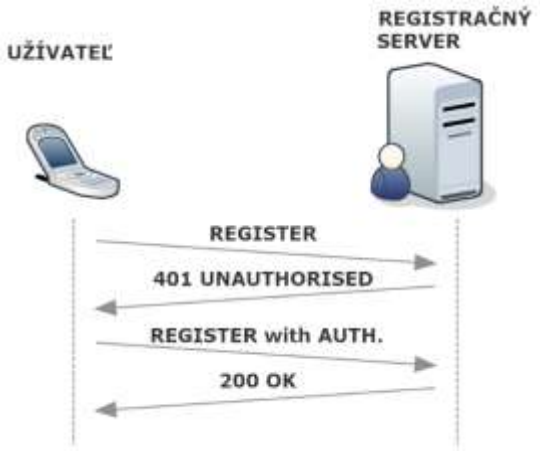
hlasovom či video prenose. SIP protokol využíva port 5060, prenáša sa primárne skrz UDP, ale dá sa využiť aj TCP. Na prenos dát sa využíva protokol RTP. Komunikácia medzi SIP zariadeniami sa vykonáva skrz žiadosti. Odpovede na SIP žiadosti sú pomocou správ s číselným kódom. SIP Proxy funguje ako stavový či bez stavový transakčný stroj. Dialóg definuje súbor správ SIP peer-to-peer medzi dvomi UA, ktoré majú vzájomnú spojitosť. Dialógy popisujú radenia a smerovanie správ medzi dvomi koncovými bodmi. SIP správy a ich obsah majú špecifickú formu, napísané sú formou textu a nadobúdajú odpovede podľa RFC 3261.

SIP je textovo založený protokol, ktorý používa dva typy signalizačných správ. Jednou je žiadosť (REQUEST) a druhou je odpoveď (RESPONSE). Výmena správ je riadená skrz transakcie. Vymieňaním týchto správ je možné vytvoriť, zmeniť či ukončiť hlasové či multimediálne spojenie, čo je definíciou SIP v RFC 3261. Správa, ktorá patrí k rovnakej transakcii a zdieľa to isté identifikačné číslo je označená v správe SIP poľom VIA BRANCH. Ilustrácia 3 zobrazuje rozdiel medzi transakciou, správou a dialógom. Na správu typu žiadosť, ktorou je INVITE, sú správy typu odpoveď TRYING, RINGING, OK prislúchajúce danej transakcii.




*Ilustrácia 3 - Zobrazenie SIP správy, transakcie a dialógu*

S systémoch s viac užívateľmi, je potrebné k tomu aj uspôsobiť prihlasovanie na server. Tak vznikne bod, ktorý slúži na sprostredkovanie spojenia, kde sa po prvý krát užívateľ prihlási do systému. Autorizáciu popisuje Ilustrácia 4.




*Ilustrácia 4 - Požiadavka o zaregistrovanie*

Protokol SIP, je nezabezpečený protokol a ktokoľvek ho môže na sieti odchytiť a analyzovať. Ilustrácia 5 zobrazuje správu INVITE zachytenú pomocou programu WIRESHARK. Správa obsahuje v riadku REQUEST-LINE URI daného servera „@192.168.30.128“, ktorého sa dožadujem nadviazania spojenia aj s menom užívateľa, ktorého sa snažím nájsť „101“. CSEQ označuje poradie žiadosti v danom dialógu, v tomto prípade sa jedná o prvú správu a jej typ, t.j. INVITE. Pole VIA označuje z akej adresy ide daná žiadosť „192.168.30.133“, jej port „5060“, pole BRANCH dáva jedinečný názov danej transakcie a dokonca Wireshark zistil aj typ agenta, od ktorého išla daná požiadavka. V poli FROM je určený dialóg označením TAG a poľom CALL-ID, ktoré tvorí dlhý reťazec znakov ukončený menom cieľovej stanice odkiaľ bola odoslaná žiadosť.

```
▼ Session Initiation Protocol (INVITE)
  ▶ Request-Line: INVITE sip:101@192.168.30.128 SIP/2.0
  ▼ Message Header
    ▶ CSeq: 1 INVITE
    ▶ v: SIP/2.0/UDP 192.168.30.133:5060;branch=z9hG4bK8428fd80-6c07-e711-9377-000c294a9996;rport
      User-Agent: Ekiga/4.0.1
    ▶ f: <sip:103@192.168.30.128>;tag=9c0dfb80-6c07-e711-9377-000c294a9996
      i: fe0ffb80-6c07-e711-9377-000c294a9996@client103-virtual-machine
      k: 100rel,replaces
    ▶ t: <sip:101@192.168.30.128>
    ▶ m: "sip:103@192.168.30.128" <sip:103@192.168.30.133:5060>
      Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,SUBSCRIBE,NOTIFY,REFER,MESSAGE,INFO,PING,PRACK
      l: 874
      c: application/sdp
      Max-Forwards: 70
  ▶ Message Body
```

*Ilustrácia 5 - Správa INVITE zachytená pomocou programu Wireshark*

Všetky tieto informácie sú v nezabezpečenej sieti a bez vhodných politík voľne prístupné a prípadný útočník sa z nich dozvie citlivé informácie o danom systéme.

Pre krátke zhodnotenie uvediem typy základných žiadostí definované v RFC 3261. Prvotnou v systéme je vždy INVITE, čím je žiadosť o zostavenie spojenia. K potvrdeniu zahájenia spojenia sa

využíva správa ACK. Pre ukončenie spojenia pri hovore je definovaná správa BYE. Ak je spojenie ukončené skorej ako bol nadviazaný hovor, posiela sa správa CANCEL. Ilustrácia 4 taktiež zobrazuje správu REGISTER, ktorá slúži pri registrovaní UA na server. Jednou z možností je aj správa OPTIONS, ktorá sa pýta servera na jeho možnosti bez toho, aby bolo zostavené spojenie.

Na každú žiadosť prichádza odpoveď. Triedy odpovedí sú tiež definované v RFC 3261 a sú doplnené číselným kódom. Sú zaradené do šiestich globálnych tried. Prvou triedou (1xx) sú dočasné informačné správy (100 TRYING, 180 RINGING); druhou triedou (2xx) sú správy s pozitívnou hodnotou ukončenia žiadosti (200 OK); v tretej triede (3xx) sú správy o presmerovaní (302 MOVED TEMPORARILY, 305 USE PROXY); obsahom štvrtej triedy správ (4xx) sú správy o chybách ktoré vznikli na strane klienta (403 FORBIDDEN); v piatej triede (5xx) sú obsiahnuté chyby, ktoré vznikli na strane servera (500 SERVER INTERNAL ERROR, 501 NOT IMPLEMENTED); a poslednou, šiestou triedou (6xx) sú správy o globálnom zlyhaní (603 DECLINE RESPONSE, 606 NOT ACCEPTABLE) [1].

Signalizácia nastáva dvomi smermi. Užívateľ odošle žiadosť a čaká na odpoveď. Ako som spomenul v sekcii 1, existuje tzv. spojovací bod, ktorý spracuje tieto požiadavky a smeruje ich podľa ich obsahu. Štandard RFC 3261 taktiež definuje typy týchto bodov, t.j. serverov, v akých režimoch pracujú, t.j. ako narábajú s danými požiadavkami. Jednou s foriem je STATEFULL server, kedy server, t.j. transakčný stroj okamžite odpovedá správou 100 TRYING na prijatie správy INVITE, čím sa potvrdí prijatie správy INVITE a zastaví sa jej prípadné opakované zasielanie. Keď server dostane odpoveď od cieľa, už neposiela jeho správu TRYING k zasielateľovi žiadosti, pretože ju už obdržal od servera. Druhou formou je server STATELESS, ktorý neodpovedá zasielateľom požiadaviek, ale čaká na odpovede od cieľa.

# 2 Architektúra navrhnutého riešenia

Architektúra všeobecne vyjadruje staviteľské umenie vytvárajúce funkčný priestor podľa ideových predstáv architekta, t.j. tvorcu pomocou dobrých technických možností. Moje staviteľstvo bude na software báze, kedy využijem dve platformy zo sveta VoIP. Je potrebné si rozviesť už skoršie spomenuté platformy a taktiež prostriedky, ktoré využijem v práci. Budú nimi, SIP proxy Kamailio, pobočková ústredňa Asterisk, kde zhodnotím ich výhody a nevýhody, ich spoločné nasadenie, rozdiely oboch platforiem a bezpečnostné prvky, ktoré dané platformy ponúkajú. Taktiež predstavím databázu MySQL, ktorá je využitá v architektúre SBC. V neposlednej rade je spomenutá architektúra SBC, jej popis a bezpečnostné prvky, ktoré aplikujem v mojej práci.

## 2.1. Úvod do SIP proxy Kamailio a pobočkovej ústredne Asterisk

### 2.1.1. SIP proxy Kamailio

SIP proxy pracuje na základe štandardu RFC 3261. Každé proxy sa rozhoduje o smerovaní žiadostí a modifikuje žiadosti predtým ako ich odošle ďalšiemu elementu v sieti, či konečnému účastníkovi. Je dôležité, aby odpovede na žiadosti postupovali tou istou cestou späť v sieti ako prichádzajúce žiadosti. SIP proxy dokáže pracovať v dvoch stavoch a to STATELESS a STATEFULL, vysvetlené v sekcii 1.2 [2].

SIP proxy Kamailio pochádza z otvorenej platformy OpenSER ktorá sa odčlenila od SER (SIP Express Router), ktorá sa v novembri 2008 opäť spojila s projektom SER a vytvorila integrovaný projekt SIP Router Project, ktorá je v dnešnej dobe vo verzii 5.0.1 (05-apríl-2017) [3]. Ja som pracoval s verziou 4.3.4, ktorá na počiatku mojej práce bola aktuálna [4].

SIP proxy Kamailio je zaštítené pod GPL licenciou, ktorý dokáže narábať s tisíckami telefonátov za sekundu. Podporuje rôzne funkcie ako, asynchrónne TCP, UDP a SCTP, zabezpečenú komunikáciu TLS, WebSocket podporu pre WebRTC, IPv4, IPv6, IM (Instant Messaging). Taktiež podporuje least cost routing, load balancing, a podporu pre databázové systémy, akými sú MySQL, Postgres, Oracle, Radius, LDAP a taktiež SNMP monitoring [5].

SIP proxy Kamailio je modulárny systém, ku ktorému je vytvorených už 193 modulov vo verzii 5.1.x (devel) [6].

Základom konfigurácie SIP proxy Kamailio je konfiguračný súbor kamailio.cfg, v ktorom sa definujú využité moduly, parametre a spôsob narábania so žiadosťami, t.j. politiky. Základné schémy politík sú obsiahnuté už v súbore kamailio.cfg. Ak je potrebné vytvoriť zložitejšiu schému práce so SIP žiadosťami, je možné využiť širokú škálu knižnice pseudopremenných a nakonfigurovať si svoje vlastné riešenie.

### 2.1.2. Pobočková ústredňa Asterisk

Tradičná pobočková ústredňa slúži ako rozhranie medzi rôznymi pripojenými stanicami, akými sú napríklad telefóny a zdrojmi, ktorými je ústredňa pripojená do vonkajšieho sveta. To znamená, že nie je možné pripojiť pobočkovú ústredňu na mieste kde sa nachádza koncová stanica a začať smerovať externé telefonáty na koncové stanice bez toho, aby užívatelia vytočili stanovenú predvoľbu pre internú či externú telefónnu sieť. Preto aj u pobočkovej ústredni Asterisk všetko čo vstúpi alebo vystupuje zo systému prechádza skrz virtuálny kanál, ktorý je obsluhovaný pomocou zoznamu telefónnych čísiel [7].

História pobočkovej ústredne Asterisk siaha do roku 1999 a jedná sa o open-source PBX. Jeho základom je jadro, ktoré sa nadväzuje na obsluhované rozhrania, akými sú moduly na spracovanie HW požiadaviek či SW požiadaviek, akým je SIP protokol. Jadro obsluhuje DIALPLAN, kde je definované správanie obsluhy požiadaviek, ktoré vstúpia so systému [7].

Konfigurácia pobočkovej ústredne Asterisk je pomocou súborov s príponou CONF. V mojej práci využijem MODULES.CONF, ktorý slúži na povolenie načítavania modulov z adresára /usr/lib/asterisk/modules. Na obsluhu SIP protokolu je využitý modul SIP.CONF, ktorý spolupracuje s modulom EXTENSIONS.CONF kde sa definuje DIAPLAN, ten obsahuje informácie ako narábať a smerovať s telefonátmi. Môj systém bude využívať prepojenie s databázovým systémom MySQL, v ktorom budú vytvorení všetci účastníci, ktorý môžu pristupovať k danému systému. Ak daný užívateľ nie je obsiahnutý v databázy, jeho požiadavka o zaregistrovanie do systému bude zamietnutá. Na prepojenie s databázou sa využije modul EXTCONFIG.CONF pomocou konektora ODBC, ktorý sa definuje v module RES_ODBC.CONF.

## 2.2. Definovanie SBC

Pojem SBC (Session Border Controler), je definovaný v RFC 5853. Jedná sa o proprietárne riešenie pre implementovanie sprostredkovateľa na sieti, ktorý pracuje s protokolom SIP a je umiestnený na hranici sieti medzi dvomi sieťami. Dôvodom je, že sieťové politiky sú zvyčajne implementované na okraji sieti. Môže ísť o dve rôzne siete poskytovateľov služby, či pripojenie siete do infraštruktúrnej siete, ale bod medzi poskytovateľom služby a sieťami zákazníkov či firiem.

Úlohou SBC je sprostredkovať určitú úroveň ochrany, akými sú skrytie topológie, ochrana pred denial of service útokmi, či poskytnúť funkcionalitu koncovým účastníkom ako preklad NAT, opravu chybným či poškodených žiadostí a odpovedí, a taktiež dostupnosť monitorovania stavoch na sieti ako je monitorovanie záťaže a kvalitu služby. SBC musí byť schopné pracovať s obidvomi signalizačnými správami, t.j. žiadosti a odpovede a taktiež musí vedieť obslúžiť aj multimediálne správy. SBC vie modifikovať hlavičky a telo správ SIP [8].

## 2.3. Kolokované nasadenie SIP proxy Kamailio a pobočkovej ústredne Asterisk v úlohe SBC

V tejto sekcii uvediem výhody a nevýhody samostatných platforiem. Vysvetlím ich úlohy, ktoré budú zohrávať v danom systéme. Na záver bude vysvetlená a zobrazená architektúra SBC ktorú budem v nasledujúcej kapitole konfigurovať.

### 2.3.1. Úloha SBC

Na úvod treba zopakovať čo je úlohou SBC. Má sa jednať o hraničný bod v sieti, ktorý sprostredkuje komunikáciu medzi žiadateľom a cieľovou stanicou. SBC je administratívny bod, ktorý kontroluje tranzit hlasových či multimediálnych elementov v komunikačnej sieti. Musí dokázať pracovať s novými aj staršími externými zariadeniami a s rôznymi sieťovými technológiami. SBC sprostredkováva smerovanie žiadostí a odpovedí podľa nastavenia v telefónnom zozname a odpovede musia vždy smerovať tou istou späť ako aj prišli do systému. SBC musí podporovať bezpečnostné politiky na vysokej úrovni. Akými sú skrytie siete, ochrana pred opakovanými telefonátmi a zasielaním opakovaného počtu rovnakých žiadostí, akými je útok distributed denial od service.

### 2.3.2. Kamailio a Asterisk

Pri spoločnom nasadení oboch platforiem je dôležité si ujasniť, ich vzájomnú spoluprácu a taktiež ich jednotlivé úlohy. Je potrebné poznať obidve platformy a ich možnosti. Je treba zvážiť, ktorá platforma bude na čo vhodnejšia. Ďalším postupom môže byť aj nahliadnutie na stránky daných platforiem a dohľadať si, aké sú momentálne dostupné moduly. Vo výsledku, ak je spoločnou kombináciou výsledná metrika nedostačujúca, je treba si zvoliť vhodnejšie moduly a preformulovať architektúru daného riešenia SBC. Poprípade upraviť využité moduly do požadovanej formy.

#### *2.3.2.1.Kamailio*

Z pohľadu RFC 3261 funguje SIP proxy Kamailio ako Registrar, Redirect Server a Proxy. Môžem ho využiť na load balancing, NAT preklad, Accounting, vo všeobecnosti sa jedná o veľmi flexibilný smerovací nástroj. SIP proxy Kamailio po prijatí správy dokáže so správnymi modulmi vykonávať inšpekciu SIP správy, upraviť príslušné časti, akými sú hlavička a telo. Na základe inšpekcie sa dokáže rozhodnúť kam ju ďalej bude smerovať, respektíve čo s ňou bude robiť. Je možné nastaviť bezpečnostné prvky, akými sú autorizácia či obmedzovať počet prijatých správ na základe rýchlosti prijatia, ich stavu, ak boli cieleným spôsobom poškodené či upravené za účelom získavania informácii alebo zaťažovania serveru za účelom jeho odstavenia. Taktiež SIP proxy Kamailio dokáže preklad protokolov.

Kamailio nie je B2BUA. Je to len Proxy server ktorý ďalej posiela dané požiadavky, Aplikačný server, média server ale dokáže pracovať s média reláciami, ale nedokáže prehrať hudbu u užívateľa, ktorý je v telefónnej fronte, pretože SIP proxy Kamailio nezaujíma ako sú dané médiá spracované. SIP proxy Kamailio je len na spracovanie signalizácie.

#### *2.3.2.2.Asterisk*

Preto je existuje dôvod na využitie ešte iného nástroja, a tým je pobočková ústredňa Asterisk. Ak chcem vytvoriť len malý systém, v princípe je postačujúca pobočková ústredňa Asterisk, kde sa vymení pôvodný hardware za softwarové riešenie Asterisk. SIP proxy Kamailio príde na rad, keď treba daný systém rozšíriť, tzn., že je potrebné implementovať load balancing či aplikovať rôzne bezpečnostné prvky.

Taktiež existujú limity aj na strane pobočkovej ústredne Asterisk. Ak príde hovor do pobočkovej ústredne Asterisk, je možné na základe EXTENSIONS.CONF ho smerovať či prehrať hudbu na počkanie. Má funkcie ako SIP ADD HEADER, aby sme získali hodnotu z danej správy, čo funguje pre počiatočnú správu INVITE. Nefunguje to pre RE-INVITE správy alebo BYE požiadavku, tzn. nedokáže pristúpiť k hodnotám BYE správ.

Nedokáže pracovať vo forme STATELESS. Vždy keď je vytvorená inštancia, odpovedá a taktiež preposiela danú požiadavku ďalej. Všetky požiadavky sú spracovávané serverom, tzn, zaťažuje ho viac ako by bolo potrebné. Taktiež tým ihneď dáva najavo potencionálnemu útočníkovi, že je funkčný a pripravený spracovávať ďalšie požiadavky. U load balancing je to negatívna vlastnosť lebo len chceme preposlať danú požiadavku na určený server, ktorý nie je momentálne vyťažený.

#### *2.3.2.3.Porovnanie platforiem*

Kompenzovať negatívnu vlastnosť u pobočkovej ústredni Asterisk, treba niečím rýchlym a malé, akým je SIP Proxy Kamailio. Ako som spomenul, pobočková ústredňa Asterisk nezvláda inšpekciu a manipuláciu SIP správ na nižšej vrstve. Ak chcem mať multi-homed sieť, kde je potrebné počúvať viacej portov a nielen 5060, je potrebné využiť SIP proxy Kamailio, lebo CHAN_SIP u pobočkovej ústredni Asterisk dokáže počúvať na jednom porte. Taktiež SIP proxy Kamailio sa dá využiť ako prezenčný server.

Pri návrhu SBC sa treba na SBC pozerať ako na hraničný bod v sieti (Single Ingress Point), ktorý slúži ako vstupný bod, cez ktorý prechádza všetka SIP komunikácia do mojej siete. Tzn., že SIP proxy Kamailio rozhoduje o žiadostiach a na základe metrík ich prepošle ďalej na pobočkovú ústredňu Asterisk, alebo ich zablokuje alebo odignoruje.

V pobočkovej ústredni Asterisk je konfigurácia na vyššej vrstve, akými je EXTENSIONS.CONF, ktorý preposiela správy na základe nastavenej metriky, potom je tam aplikácia RINGING, ANSWER, PLAYBACK, VOICEMAIL, kde sa dá definovať postup spracovania danej požiadavky, t.j. v jednoduchosti povedané, telefonátu.

#### *2.3.2.4.Kamailio s bezpečnostnými prvkami SBC*

V SIP proxy Kamailio som na úrovni SIP správy, takže sa nedá odpovedať na daný telefonát zdvihnutím. Dá sa vykonávať inšpekcia, o aký typ SIP správy sa jedná, od koho, aké má hodnoty a čoho sa dopytuje. Ako som spomenul, v tomto bode je možné upraviť hlavičku, či určité časti z danej správy a preposlať danú žiadosť, telefonát.

BFA (Brutal Force Attack) sa snaží pomocou posielania REQUEST a INVITE správ, zistiť slabiny na pobočkovej ústredni Asterisk, akými sú slabé heslá užívateľov. Nato sa využíva vonkajší SW, ktorým je napr. fail2ban, kde blokuje danú opätovnú požiadavku na základe IP adresy. Ale prečo neblokovať na základe položky User-agent v SIP správe, kde už môže byť indícia k tomu kto stojí za daným útokom. Túto možnosť pobočková ústredňa Asterisk nemá. Preto znova do hry vstupuje SIP proxy Kamailio, ktorý dané požiadavky na bezpečnosť dokáže sprostredkovať využitím správnych modulov. Naďalej bude možné prijímať správy z danej IP adresy a odignoruje všetky so zlými hodnotami v user-agent.

Ako som spomenul, so SIP proxy Kamailio je možné pracovať, manipulovať so SIP správou; t.j. odoberať hlavičky, pripájať hlavičky, upravovať hlavičky; napríklad chcem skryť určitú časť hlavičky, ktorú nechcem prezentovať na ceste ďalej v poli VIA v správe SIP.

So SIP proxy Kamailio je možné zaznamenávať informácie o telefonátoch, akým je ich dĺžka, kvalita, jitter, oneskorenie do databázy alebo ich zapísať do hlavičky SIP správy. Dáta je možné poskytnúť vo forme grafov či iného výstupu zákazníkovi, ktorý si vizuálne overí funkčnosť a kvalitu produktu, za ktorý si zaplatil. Táto možnosť odpadá s pobočkovou ústredňou Asterisk, lebo nevie pristupovať k hlavičkám SIP správ.

### 2.3.3. Kamailio a jeho úloha

So SIP proxy Kamailio sa dá zabrániť útokom, ktoré sú za účelom skenovania a zistenia siete, alebo zabrániť už registrovaným užívateľom zaplaviť danú sieť požiadavkami, ktoré zisťujú mená a heslá užívateľov v systéme. V SIP proxy Kamailio je toto všetko možné zabezpečiť.

Možnosťami je buď statické blokovanie, to je na základe IP adresy, užívateľského mena; alebo dynamické blokovanie, ktoré je založené na počte žiadostí určitého typu, ako zdrojová IP adresa, zdrojová sieť, typ požiadavky, číslo či meno zdrojového užívateľa. Naskytuje sa tu možnosť tzv. false-positives, keď klient môže odosielať priveľa žiadostí a bude na tom základe zablokovaný. Taktiež ak budú limity na zablokovanie príliš nízke, zablokuje sa aj právoplatný užívateľ.

SIP proxy Kamailio má veľa modulov, ktoré sa dajú využiť na vyššie spomenuté techniky zabezpečenia a ochrany. Ja budem aplikovať ochranu skrytia topológie siete. Taktiež aplikujem ochranu pred zaplavením siete požiadavkami, vytvorím ochranu proti brutal force attack. Vypracoval som formu zabezpečenia pred distributed denial of service útokom a implementoval som zabezpečenú formu výmeny komunikácie.

Moduly, ktoré využijem v SIP proxy Kamailio sú TOPOH, na skrytie topológie siete. Ochranu pred zaplavením správami (Flood protection) zabezpečím pomocou modulov HTABLE a PIKE. Modul HTABLE slúži k zaznamenaniu klientov, t.j. ak sa vyskytne požiadavka, on ich zapisuje do virtuálnej hash tabuľky, podľa ktorej je možné vedieť či už daný užívateľ či IP adresa je už blokovaná alebo má dovolené posielať ďalšie žiadosti. PIKE modul prepočítava či daná správa je v rámci povoleného limitu žiadostí za určitý časový interval. Ochranu pred brutal force útokom je využitý modul HTABLE a správna kombinácia autorizácie. Na ochranu pred DDoS útokmi je zabezpečená implementáciou modulu RATELIMIT, ktorý limituje blokáciu na základe intenzitu prichádzajúcich SIP žiadostí. V tom sa modul líši od modulu PIKE, ktorý sa aplikuje len na IP adresu.

### 2.3.4. Asterisk a jeho úloha

Slabinou v mojom systéme je slepá dôvera pobočkovej ústredne Asterisk ku SIP proxy Kamailio. Jedná sa o preberanie informácií bez akejkoľvek kontroly. Preto treba dbať na bezpečnosť a kontrolovať aké požiadavky budú smerované k pobočkovej ústredni Asterisk.

Hlavnou úlohou pobočkovej ústredne Asterisk v mojom systéme bude spojenie s internou sieťou a komunikácia s databázou užívateľov v MySQL pomocou konektora ODBC.

### 2.3.5. MySQL a jeho úloha

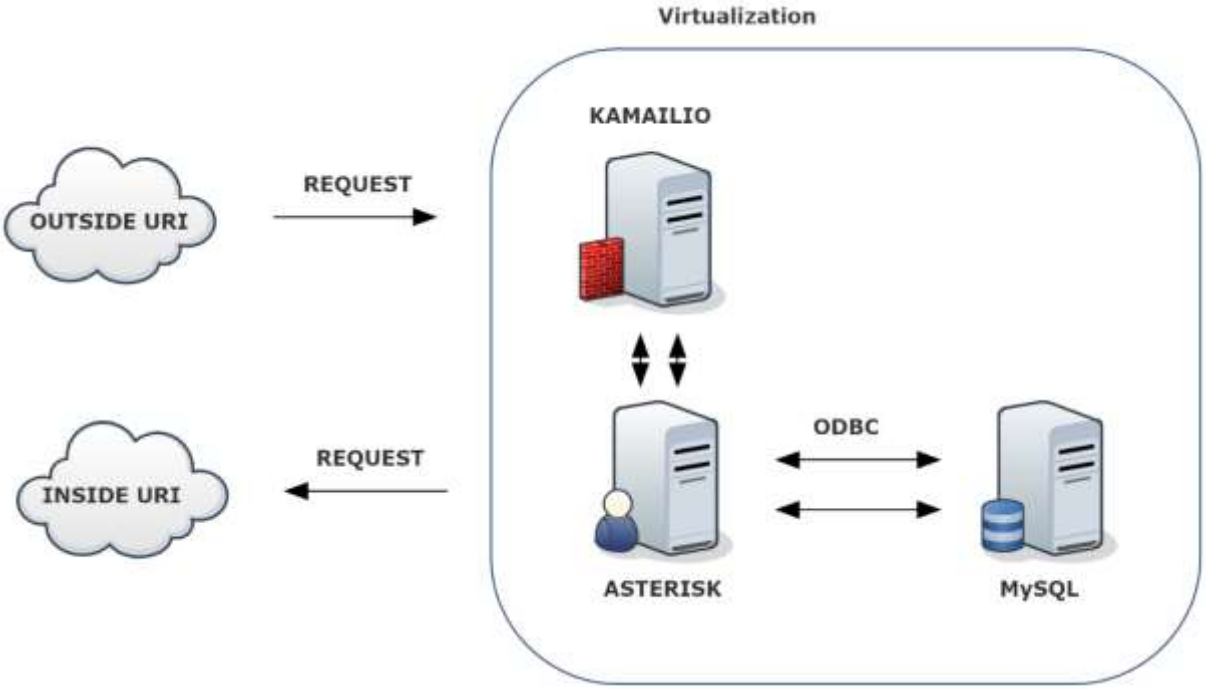
Do komunikačného systému nemôže byť povolené vstúpiť každému. Chcem, aby sa dokázali prihlásiť len známi užívatelia. Riešením je vytvoriť tabuľku respektíve databázu užívateľov, ktorí budú overovaní prihlasovacím menom a heslom pri prihlasovaním do systému, t.j. budú autentifikovaný a na základe autentifikácie im bude povolená autorizácia vykonávať telefonáty s ďalšími užívateľmi.

Ako vhodný systém na spravovanie užívateľov je databázový systém. V mojom riešení som využil databázový systém MySQL, ktorý používa na prácu s databázovým prostredím jazyk SQL.

## 2.4. Architektúra realizovaného SBC

Koncept SBC sa opiera o tri základné piliere. Prvým je administratívna hranica, ktorá aplikuje bezpečnostné politiky a smeruje požiadavky na ciele podľa rozhodnutia telekomunikačného bodu. Ňou je SIP proxy Kamailio. Druhým je telekomunikačný bod, ktorý spracováva požiadavky na overenie užívateľov a poskytnutia URI cieľa, ktorého sa dožaduje administratívna hranica. Túto obsluhu má na starosti pobočková ústredňa Asterisk. A treťou časťou je užívateľský bod, kde sú zaznamenaní užívatelia, ktorí majú autorizáciu na vykonávanie telefonátov. Tým je databáza vytvorená v prostredí MySQL.

Vyššie spomenutý koncept práce jednotlivých častí v SBC vystihuje Ilustrácia 6.




*Ilustrácia 6 - Architektúra prostredia*

# 3 Implementácia spoločného nasadenia

Využil som nasledovné verzie platforiem:

**Ubuntu 16.04.1 (Xenial Xeros)**

**Asterisk 13.1.0**

**Kamailio 4.3.4-1.1ubuntu2 (kamctl 4.2.0)**

## 3.1. Inštalácia prostredia Kamailio, Asterisk MySQL a ODBC

Začnem vytvorením virtuálneho stroja v prostredí VMware. Kroky k vytvoreniu sú uvedené v prílohe na strane I. V krokoch sú uvedené informácie o veľkosti RAM a HDD či počet CPU, ktoré využijem vo svojom virtuálnom prostredí.

Inštalácia operačného systému Ubuntu Server na virtuálny stroj je rozpísaný v prílohe na strane I. Jeho obsahom sú kroky ako výber jazyka, klávesnice, nastavenie času podľa časovej zóny a ku koncu sa vyberá typ manažéra logickej jednotky na diskovom poli a či chcem niektoré balíčky už nainštalovať rovno z repozitára umiestneného na DVD nosiči.

Nainštalujem pobočkovú ústredňu Asterisk, SIP proxy Kamailio, MySQL-server a MySQL-klienta. Využije sa repozitár umiestnený na internete a pred inštaláciou je vhodné si aktualizovať cesty na repozitáre a taktiež si aktualizovať stávajúce nainštalované balíčky na systéme. Nato sa využije príkaz INSTALL UPDATE && UPGRADE, po ktorom je vhodné si daný systém reštartovať príkazom REBOOT.

```
# apt-get install asterisk
# apt-get install kamailio
# apt-get install mysql-server mysql-client
```

K vytvoreniu prepojenia medzi pobočkovou ústredňou Asterisk a databázovým systémom MySQL je potrebné použiť ODBC konektor [9]. Prvé dva balíčky nainštalujem rovno z repozitára, ale balíček libmyodbc nemá aktuálnu verziu pre MySQL verzie 5.7.0 (ktorú som nainštaloval) náhradu, preto musím siahnuť po náhrade z verzie 5.2.0 [10] [11]. Treba stiahnuť zo stránky, skopírovať pomocou winscp na server do zložky home/student odkiaľ ho rozbalím a skompilujem.

```
# apt-get install unixodbc-dev
# apt-get install libmysqlclient-dev
# apt-get install libmyodbc
Download: mysql-connector-odbc-5.3.6-linux-ubuntu15.10-x86-
64bit.tar.gz
location: https://dev.mysql.com/downloads/connector/odbc/
winscp (from local)
# cd /home/student/
# tar -xvf mysql-connector-odbc-5.3.6-linux-ubuntu15.10-x86-
64bit.tar.gz
```

```
# cp mysql-connector-odbc-5.3.6-linux-ubuntu15.10-x86-
64bit/lib/libmyodbc5* /usr/lib/x86_64-linux-gnu/odbc/
# /home/student/mysql-connector-odbc-5.3.6-linux-ubuntu15.10-
x86-64bit/bin/myodbc-installer -d -a -n "MySQL" -t
"DRIVER=/usr/lib/x86_64-linux-gnu/odbc/libmyodbc5w.so;"
```

## 3.2. Konfigurácia vzájomného prepojenia

Ďalším postupom je nastavenie ODBC konfiguračného súboru a konektora [12] [13] [14]. Konfigurácie pre ODBC konektor sa nachádzajú v prílohe C strana II. V ODBCINST:INI sa definuje meno ovládača, cesta pre ovládač, ktorý bude pracovať s MySQL a počet možných súčasných relácií. Pre ODBC.INI je konfigurácia zložitejšia, lebo treba správne definovať prístupové meno a heslo pre databázu pobočkovej ústredni Asterisk v MySQL, meno databáze, využívaný port na komunikáciu a typ soketu.

```
# nano /etc/odbcinst.ini
# nano /etc/odbc.ini
```

Na strane pobočkovej ústredne Asterisk potrebujem nastaviť spojenie k databáze pomocou ODBC, kde som vymazal celý obsah a začal odznova vypĺňať potrebné časti [15]. Konfigurácia súboru RES_ODBC.CONF pre pobočkovú ústredňu Asterisk sa nachádza v prílohe D strana II. Obsahom konfigurácie je meno spojenia, či je povolené spojenie, či sa môžem ihneď pripájať k danej databáze, časová dilatácia pri pripojení, limit pripojení v rovnakom čase, s akým menom je možné sa na dané pripojenie odkazovať, meno databázy a prístupové meno a heslo k danému pripojeniu.

```
# nano /etc/asterisk/res_odbc.conf
```

Ďalším krokom je vytvorenie databázy a tabuliek pre pobočkovú ústredňu Asterisk v databázovom prostredí MySQL. Keďže sa jedná o zložité a dlhé definície tabuliek, ich štruktúru som neuviedol vo svojej práci. Tabuľky sú prístupné na odkaze [16], odkiaľ je možné ich skopírovať a využiť pri ich vytvorení. Uvediem len zoznam tabuliek, ktoré je potrebné mať v mojom systéme. Vytvárajú sa príkazom CREATE TABLE. Tabuľky sú, SIPREGS, SIPUSERS, VOICEMAIL, VOICEMAIL_DATA a VOICEMAIL_MESSAGES.

http://kb.asipto.com/asterisk:realtime:kamailio-4.0.x-asterisk-11.3.0-astdb

```
# mysql -u root -p
# CREATE DATABASE asterisk;
# USE asterisk;
# GRANT ALL ON asterisk.* TO asterisk@localhost IDENTIFIED BY
'asterisk_password';
# CREATE TABLE ...
```

Otestujem si funkčnosť pripojenia ODBC k databáze MySQL.

```
# isql -v MySQL-asterisk
```

```
+--------------------------------------+
| Connected!                           |
|                                      |
| sql-statement                        |
| help [tablename]                     |
| quit                                 |
|                                      |
+--------------------------------------+
```

*Ilustrácia 7 - Otestovanie spojenia medzi databázou MySQL a pobočkovou ústredňou Asterisk*

Keďže chcem využívať overovanie užívateľov v reálnom čase pre pobočkovú ústredňu Asterisk, je potrebné nakonfigurovať súbor EXTCONFIG.CONF. Jeho obsah sa nachádza v prílohe D strana III. Jeho obsahom sú názvy tabuliek, typ konektora, ktorý sa bude pripájať, meno a heslo.

```
# nano /etc/asterisk/extconfig.conf
```

Overovanie v reálnom čase sa nastavuje v konfiguračnom súbore SIP.CONF. Jeho obsahom je ešte typ protokolu, aký sa použije na prenos, port a IP adresa, ktorá sa má počúvať. Úplná konfigurácia sa nachádza v prílohe D strana III.

```
# nano /etc/asterisk/sip.conf
```

Keďže som už viackrát spomenul, že pobočková ústredňa musí pracovať so zoznamom volaných čísiel, u platformy Asterisk ním je súbor EXTENSIONS.CONF. V ňom si vytvorím DIALPLAN, ktorý budem v mojom systéme využívať. Obsahom je schéma vytáčaných čísiel. Jeho úplný obsah je dostupný v prílohe D strana III

```
# nano /etc/asterisk/extensions.conf
```

Taktiež sa vyskytli problémy pri štarte pobočkovej služby Asterisk a bolo treba niektoré služby (moduly) vypnúť a iné doplniť [17] [18]. Nato slúži konfiguračný súbor MODULES.CONF, kde sa definujú moduly pobočkovej ústredne Asterisk a spôsob ich načítania pri štarte služby. Moja konfigurácia modulov je uvedená v Prílohe D strana III

```
# nano /etc/asterisk/modules.conf
```

Po zvládnutí konfigurácie pobočkovej ústredne Asterisk, treba upraviť aj tabuľky ktoré boli vytvorené v databázovom prostredí MySQL. Prvým krokom je pridanie užívateľov do tabuľky SIPUSERS pomocou príkazu INSERT INTO a s hodnotami, akými sú meno, heslo, telefónne číslo a IP adresa servera. Zoznam príkazov sa nachádza v Prílohe E strana IV.

```
# mysql -u root -p
# use asterisk;
# INSERT INTO ...
```

Keďže pri spustení služby Asterisk sa vyskytli problémy so zápisom do polí USERAGENT, kedy nebolo možné prideliť dáta do bunky LOCALSETS, kvôli svojej obmedzenej veľkosti bolo treba rozšíriť dané pole. Príkazy na úpravu sú uvedené nižšie.

```
# update sipusers set context='LocalSets';
# alter table sipusers modilfy useragent varchar(255);
# use kamailio;
# alter table sipusers modilfy useragent varchar(255);
# quit;
```

Po nasledovných úpravách je potrebné reštartovať modifikované služby.

```
# service mysql restart
# service asterisk restart
```

Ak boli správne nakonfigurované ODBC súbory sa dajú overiť v CLI pobočkovej ústredne Asterisk. Príkazom ODBC SHOW zistíme pripojenie pomocou ODBC konektoru, ktoré v mojom prípade zobrazuje Ilustrácia 8.

```
# asterisk -r
# odbc show

ODBC DSN Settings
-----------------
  Name:   asterisk
  DSN:    MySQL-asterisk
    Last connection attempt: 1970-01-01 01:00:00
  Pooled: No
  Connected: Yes
```

*Ilustrácia 8 - Test funkčnosti ODBC konektora*

Úpravu platformy Asterisk mám hotovú a pristúpim teraz k úprave a konfigurácii na SIP proxy Kamailio. Prvotne treba vytvoriť správne databázu pre MySQL, upravím nasledovné riadky v konfigurácii KAMCTLRC a taktiež pridám pre OSER_FIFO správne nastavenie [19] [20] [21]. Jeho úlohou je definovanie prístupu do databázy, ktorá bude vytvorená v prostredí MySQL a jeho obsahom sú užívateľské mená a heslá na server s danou IP adresou. Jeho úplný obsah sa nachádza v Prílohe F na strane V.

```
# nano /etc/kamailio/kamctlrc
```

Po úprave KAMCTLRC je možné vytvoriť databázu, ktorú bude využívať SIP proxy Kamailio.

```
# /usr/sbin/kamdbctl create
```

Teraz nadišiel čas po prvýkrát upraviť moje SIP proxy Kamailio v konfiguračnom súbore KAMAILIO.CFG, ktorý je alfa i omega všetkého ako Kamailio spracováva žiadosti, t.j. je v ňom zahrnutá smerovacia a zabezpečovacia politika.

Treba zadefinovať základné moduly, ktoré budú pracovať v SIP proxy Kamailio, akými sú MySQL, Autorizácia, Asterisk. Nastaviť porty, cez ktoré nastane výmena informácií medzi SIP proxy Kamailio (port 5060) a pobočkovou ústredňou Asterisk (port 5080). Detailná konfigurácia sa nachádza v Prílohe G na strane V.

```
# nano /etc/kamailio/kamailio.cfg
```

Po úspešnom nastavení spustím SIP proxy službu Kamailio pomocou KAMCTL START. Po spustení sa zobrazí informácia o tom, že moduly pre MySQL a FIFO frontu sú spustené, a pod akým procesným identifikačným číslom sa služba Kamailio spustila. Názorný príklad zobrazuje Ilustrácia 9.

```
# kamctl start

database engine 'MYSQL' loaded
Control engine 'FIFO' loaded
INFO: Starting Kamailio :
INFO: Started (pid: 19426)
```

*Ilustrácia 9 - Test funkčnosti Kamailia*

Službu Kamailio je možné sledovať pomocou príkazu KAMCTL MONI, ktorý zobrazuje nahrané moduly, ktoré sú zadefinované v konfiguračnom súbore KAMAILIO.CFG, dĺžku spustenej služby a stručný štatistický prehľad o žiadostiach, ktoré sa vyskytli v systéme. Ilustrácia 19 v Prílohe H na strane VI, zobrazuje taký výpis.

```
# kamctl moni
```

### 3.2.1. Overenie funkčnosti prostredia

Celý čas počas konfigurácie a testovania som využíval dva oporné body. Jedným z nich bol softphone Linphone a nástroj pre zachytávanie komunikácie na sieti Wireshark. Inštalácia a nastavenie programu Linphone sa nachádza v prílohe I na strane VII. Program Wireshark je popísaný v prílohe J na strane VII. K tomu bolo vhodné si vytvoriť a povoliť zapisovanie správ do prostredia SYSLOG pre prehľadnejšie zobrazenie statusu z bezpečnostných metrík. Nato som využil modul XLOG, ktorý je prístupný na stránkach kamailio.org v sekcii moduly.

Na nastavenie modulu XLOG a sprístupnenie logovania do SYSLOG bolo treba aktivovať modul a veľkosť logovania v KAMAILIO.CFG [22] [23].

```
# nano /etc/kamailio/kamailio.cfg

#!define WITH_DEBUG

#!ifdef WITH_DEBUG
debug=4
log_stderror=yes
#!else
debug=2
log_stderror=no
#!endif

loadmodule "xlog.so"
```

```
# ----- xlog params -----
modparam("xlog", "buf_size", 8192)
modparam("xlog", "force_color", 1)
```

Pre vhodný výpis treba nastaviť správne parametre, t.j. reťazce, pre filter GREP. Dá sa využiť parameter ako správa SIP či PIKE či inú hodnoty ako IP adresa či meno užívateľa a v neposlednej rade sa dá využiť aj iná hodnota akou je hodnota poľa USER-AGENT.

```
# tail -f /var/log/syslog |grep sip
```

# 4 Bezpečnostné prvky realizované v SBC

Po implementácii základného funkčného prostredia je teraz možné pristúpiť ku skoršie spomenutým bezpečnostným praktikám, ktoré budú plniť úlohu SBC.

## 4.1. Integrácia prvkov SBC

### 4.1.1. Topology hiding TOPOH

Tento modul skrýva SIP hlavičky, ktoré zobrazujú topológiu siete. Nezáleží na tom či je server Stateless alebo Statefull. TOPOH Skript dekóduje SIP hlavičku, a celá doteraz nastavená funkcionalita ostáva nezmenená.

Konfigurácia modulu nie je zložitá, jedná sa o pár príkazov a aj jeho implementácia nezaberá veľa času. TOPOH môže byť nasadený do živého prostredia, dokonca aj počas existujúcich telefonátov. Stačí len reštartovať SIP proxy Kamailio a TOPOH začne ihneď pracovať. Nijaký telefonát nebude zahodený a stratený.

Vďaka využitiu hodnoty MASK-KEY, viaceré SIP servery sú stále schopné dekódovať hlavičku správne. Využitie tejto funkcionality je najmä u load-balance [24].

#### *4.1.1.1.Implementácia*

Upravím KAMAILIO:CFG o načítanie TOPOH modulu s nasledovnými parametrami, kľúč, ktorý sa využíva na maskovanie siete a IP adresu, ktorá bude využitá na miesto skutočnej IP adresy.

Predispozíciou je načítanie modulu RR.SO.

```
loadmodule "topoh.so"

# ----- topoh params ----
modparam("topoh", "mask_key", "123456")
modparam("topoh", "mask_ip", "10.1.1.10")
```

#### *4.1.1.2.Testovanie*

Po načítaní a nastavení modulu TOPOH s parametrami v sekcii 4.1.1.1, je možné vidieť zmenu IP adresy z internej IP adresy na takú, ktorú chceme publikovať ďalej.

Rozdiel je možné pozorovať v poli CONTACT ktorý sa líši IP adresami. Celý výpis z INVITE správy sa nachádza v prílohe A a B na strane VIII.

```
IN: Contact: <sip:102@192.168.30.132>
OUT: Contact: <sip:10.1.1.10>
```

### 4.1.2. Flood protection PIKE & HTABLE

#### *4.1.2.1.Modul PIKE*

PIKE modul zaznamenáva všetky prichádzajúce IP adresy a zablokuje tie, ktoré prekračujú limit. Pracuje s IPv4 a aj IPv6 adresami.

Tento modul neposkytuje službu vo forme zablokovania, jeho úlohou je len vytvorenie záznamu o tom, že niekde sa zvyšuje prítomnosť požiadaviek na danej IP adrese. Je len na rozhodnutí administrátora ako s danými záznamami bude ďalej pracovať.

Modul nemá žiadne predispozície [25].

#### *4.1.2.2.Modul HTABLE*

Tento modul pridá kontajner hash table ku konfiguračného jazyku. Hash table je uložený v zdieľanej pamäti a je možné k nej pristupovať skrz pseudopremenné. Modul podporuje viacero hash tables a dokáže načítavať dáta po štarte z databázy.

Typickým prípadom implementácie na SIP serveri je vytvorenie cache systému v konfiguračnom súbore. V definícii tabuľky sa dá definovať expiračný čas načítaných dát.

V mojom prípade, žiadne predispozície na čítanie daného modulu [26].

#### *4.1.2.3.Implementácia*

Treba definovať modul ANTIFLOOD s moduly PIKE a HTABLE.

Nasledovne prejdem k definícii PIKE modulu. Sampling time unit určuje ako často bude odoberať vzorky. Čím menšie číslo, tým častejšie odoberá vzorky. Na zistenie výkyvov, t.j. špičiek treba zvoliť malú hodnotu. Malá hodnota, ale vyžaduje viacej záťaže na strane obsluhy. Regs density per unit udáva koľko požiadaviek bude povolených na sampling time unit predtým ako bude zablokované prijímanie žiadosti z danej IP adresy. Odporúčaná hodnota je pre IPv4 3x hodnota sampling time unit. Remove latency špecifikuje ako dlho ostane daná IP adresa v pamäti po jej poslednej požiadavke. Je definovaná v sekundách [27] [28] [29] [30].

U HTABLE modulu si definujem veľkosť hash tabuľky v parametri size. Autoexpire je čas v sekundách, po ktorom bude daný záznam z tabuľky vymazaný.

```
#!define WITH_ANTIFLOOD

#!ifdef WITH_ANTIFLOOD
loadmodule "htable.so"
loadmodule "pike.so"
#!endif
```

```
#!ifdef WITH_ANTIFLOOD

# ----- pike params -----
modparam("pike", "sampling_time_unit", 2)
modparam("pike", "reqs_density_per_unit", 20)
modparam("pike", "remove_latency", 4)

# ----- htable params -----
# ip ban htable with autoexpire after 5 minutes
modparam("htable", "htable", "ipban=>size=8;autoexpire=300;")
#!endif
```

#### *4.1.2.4.Testovanie*

Výsledný záznam je dostupný v SYSLOGu. Je jeho skrátený záznam je zobrazený nižšie. Pole HITS zobrazuje hodnotu koľkýkrát bola daná hodnota zapísaná do tabuľky, pričom splnila parametre. Hodnota 4 je maximum na základe reqs density per unit a FUNC_FLAGS s hodnotu 6 značí, že od posledného záznamu je daná IP adresa zablokovaná, viď tretí záznam.

```
Mar 27 17:55:29 server kamailio[107623]: 8(107635) DEBUG: pike
[pike_funcs.c:89]: pike_check_req(): src IP [192.168.30.138],
node=0x7f97f2f3ec08; hits=[0,0],[0,4] node_flags=14 func_flags=6

Mar 27 17:55:29 server kamailio[107623]: 8(107635) WARNING: pike
[pike_funcs.c:151]: pike_check_req(): PIKE - BLOCKing ip
192.168.30.138, node=0x7f97f2f3ec08

Mar 27 17:55:29 server kamailio[107623]: 8(107635) ALERT: <script>:
ALERT: pike blocking INVITE from sip:student@192.168.30.128
(IP:192.168.30.138:5060)
```

### 4.1.3. Brutal Force Attack protection HTABLE

Doterajšiu konfiguráciu som rozšíril v sekcii htable params o nový riadok, ktorý slúži na zablokovanie užívateľa, ktorý sa snaží autorizovať so zlými heslami. K tomu treba upraviť aj pôvodný route[REQINIT] [31].

```
# ----- htable params -----
modparam("htable", "htable", "userban=>size=8;autoexpire=220;")
#!endif

route[REQINIT]
...
if(is_present_hf("Authorization"))
{
```

```
if($sht(userban=>$au::auth_count)==3)
{
$var(exp) = $Ts - 200;
if($sht(userban=>$au::last_auth) > $var(exp))
{
sl_send_reply("403", "Try later");
exit;
} else {
$sht(userban=>$au::auth_count) = 0;
}...
```

### 4.1.4. Distributed Denial of Service Attack protection RATELIMIT

Možnosťami ako sa chrániť proti DDoS je v prostredí SIP proxy Kamailio viacero. Možnosťou je kombinácia PIKE, HTABLE, RATELIMIT alebo PIPELIMIT. Ja som si zvolil pracovať s RATELIMIT modulom.

Modul RATELIMIT implementuje limit intenzity prichádzajúcich SIP žiadostí do systému. Na rozdiel od modulu PIKE, RATELIMIT limituje prevádzku na základe SIP žiadostí a nie na základe IP adresy zdroja.

Ako predispozícia pred použitím musí byť sprístupnený SL modul [32].

#### *4.1.4.1.Implementácia*

Dokiaľ nie je naplnený limit, odpoveďou na všetky prijaté požiadavky je OK správa. Ak prekročia žiadosti stanovené limity, skrz RL_CHECK je možné neodpovedať, t.j. SIP server sa bude správať v STATELESS režime.

Počiatočná dĺžka časovača v sekundách sa nastaví v timer interval. Množstvo správ treba podeliť týmto číslom, aby som dostal správy za sekundu, Malá hodnota môže ovplyvniť výkon. Pomocou queue definujem fronty pre každý typ správ osobitne a pomocou PIPE im pridelím aj obmedzenie [33].

```
loadmodule "ratelimit.so"

# ----- ratelimit params -----
modparam("ratelimit", "timer_interval", 5)
# assign pipe no 0 to method REGISTER
# assign pipe no 3 to method INVITE
# assign pipe no 2 to all other methods
modparam("ratelimit", "queue", "0:REGISTER")
modparam("ratelimit", "queue", "3:INVITE")
modparam("ratelimit", "queue", "2:*")
```

```
# define pipe 0 with a limit of 200 pkts/sec using TAILDROP
algorithm
# define pipe 1 with a limit of 100 pkts/sec using RED algorithm
# define pipe 2 with a limit of 50 pkts/sec using TAILDROP algorithm
# define pipe 3 with a limit of load factor 80 using FEEDBACK
algorithm
# define pipe 4 with a limit of 10000 pending bytes in the rx_queue
using NETWORK algorithm
modparam("ratelimit", "pipe", "0:TAILDROP:200")
modparam("ratelimit", "pipe", "1:RED:100")
modparam("ratelimit", "pipe", "2:TAILDROP:50")
modparam("ratelimit", "pipe", "3:FEEDBACK:80")
modparam("ratelimit", "pipe", "4:NETWORK:10000")
```

Nasanie rl_check [34] [35].

```
}
if (is_method("INVITE|REGISTER|SUBSCRIBE")) {
if (!rl_check()) {
xlog("L_INFO","INFO: ratelimit was reached $rm from $fu (IP:$si:$
append_to_reply("Retry-After: 5\r\n");
sl_send_reply("503","Limiting");
exit;
};
};
```

### 4.1.5. TLS

Modul TLS implementuje využitie TLS transportu pomocou OpenSSL knižnice. Základom je aktivovanie TLS modulu a nastavení správnych parametrov. K správnemu nastaveniu TLS je treba mať certifikát aj privátny kľúč [36].

Definujem TLS modul.

```
#!define WITH_TLS
```

Nastavím počúvanie pre TLS port, ktorý som si určil ako 5061.

```
listen=tls:192.168.30.128:5061
```

Povoliť využívanie TLS.

```
#!ifdef WITH_TLS
enable_tls=yes
#!endif
```

Načítam modul.

```
#!ifdef WITH_TLS
loadmodule "tls.so"
#!endif
```

V tomto konfiguračnom súbore sa nachádza nastavenie TLS, t.j. nastavím parametre na certifikát aj meno kľúča, ktorý bude použitý (tie vytvorím neskôr).

```
#!ifdef WITH_TLS
# ----- tls params -----
modparam("tls", "config", "/etc/kamailio/tls.cfg")
#!endif
```

Detail konfiguračného súbora tls.conf a postup vytvorenia certifikátu a privátneho kľúča sa nachádza v prílohe na strane IX a IX [37] [38] [39] [40]

# 5 Testovanie

## 5.1. SIPp

SIPp je open source testovací nástroj a generátor požiadaviek pre SIP protokol. Dokáže pracovať s vlastnými scenármi kde sa dajú definovať spôsoby vytvárania požiadaviek na server. Výsledky zobrazuje v štatistickej tabuľke kde sú uvedené počty telefonátov, doby trvania daného požiadavku, počet retransmisií a celkový čas testu. Podporuje protokol IPv6, TLS, SCTP, SIP autorizácie, UPD retransmisie, testuje robustnosť (ochranu protokolu). Taktiež dokáže posielať audio či video tok (RTP) pomocou prehrávania PCAP súboru [41].

### 5.1.1. Nasadenie SIPp

Potrebné balíčky na inštaláciu sú uvedené v prílohe na strane XI. SIPp je potrebné stiahnuť zo stránok Github [42] viď link, Použijem Winscp na skopírovanie na klienta, rozbalím a skompilujem [43].

https://github.com/SIPp/sipp/releases/download/v3.5.1/sipp-3.5.1.tar.gz

```
# mkdir sipp
# tar -xvzf sipp-3.5.1.tar.gz
# /home/student/sipp/sipp-3.5.1#
# ./configure --with-pcap
# Make
```

Teraz je potrebné vytvoriť scenáre. Prvý súbor CSV sa využíva na volanie vytvoreného scenára XML. Jeho obsahom je meno užívateľa, IP adresa servera a autentifikačné údaje užívateľa. XML definuje spôsob volania a jeho obsahom je komplexný scenár, ktorý definuje polia FROM, VIA, TO, CSEQ. Obidva konfiguračné súbory sú uvedené detailne v prílohe na strane XI a XI [44] [45]. Jedná sa o REGISTER-DATA.CSV a REGISTER.XML.

Súbory CSV a XML využijem pri vytváraní testu pomocou nižšie uvedeného skriptu. Jeho obsahom sú príznaky -SF, za ktorým sa udáva cesta ku skriptu XML scenára, príznak -INF, ktorý vloží dáta zo súboru CSV o prihlasovaní počas vykonávania scenára, príznak -M s počtom telefonátov, po ktorom sa skončí vykonávanie testu, príznak -L s údajom o maximálnom počte simultánnych telefonátov a príznak -R kde sa definuje intenzita telefonátov za jednotku času. Keďže jednotka času pomocou príznaku -RP nie je definovaná tak základnou hodnotou je jedna sekunda. Ďalej sú uvedené príznaky -I, za ktorou sa udáva lokálna IP adresa, t.j. odkiaľ je daný test spúšťaný, príznak -P, ktorý definuje lokálny port, ktorý sa využije pri teste a za ním IP adresa servera, ktorý testujeme. Voliteľné sú taktiež príznaky na vytvorenie záznamov, akými sú -TRACED_STAT, -TRACE_ERR, -TRACE_LOG a -TRACE_COUNTS, ktoré som využil pri každom teste.

Základná logika pre vytvorenie testu sa nachádza nižšie.

```
./sipp -sf <file>.xml -inf <file>.csv -m <počet hovorov> -l
<pocet súbežných hovoru> -r <koľko hovorov za sekundu> -i <IP> -p
<port> -trace_stat -trace_err -trace_log
```

## 5.2. Výkonnostné testovanie

U výkonnostného testovania sa budem opierať o testovací nástroj SIPp a využívať vytvorené scenáre a skript na spustenie viď podkapitola 5.1.1. Priradením testovacích parametrov vznikne nižšie uvedený test kde simulujem 5000 vykonaných telefonátov, pričom bude 10000 súbežných požiadaviek po 200 telefonátoch za sekundu na cieľový port 5060 s IP adresou 192.168.30.128, čo je adresa môjho SBC.

```
./sipp -sf register.xml -inf register-data.csv -m 5000 -l 10000
-r 200 -i 192.168.30.131 -p 5060 192.168.30.128 -trace_stat -
trace_err -trace_logs -trace_counts
```

Výsledkom je výpis z testu ako som spomenul v podkapitole 5.1.1. Ilustrácia 1Ilustrácia 10 zobrazuje len stručný výpis z testu, ktorý obsahuje len počet úspešných a neúspešných požiadaviek. Celý výpis sa nachádza v prílohe R na strane XII .

```
Successful call        |        0                  |         3
Failed call            |        0                  |      4997
```

*Ilustrácia 10 - Výňatok z uceleného výpisu po testovaní pomocou programi SIPp*

Celkovo som vytvoril dvanásť testovacích scenárov, pri ktorých sa menili tri parametre zabezpečenia SBC. Parametrami boli sampling time unit, ktorý hovorí o tom, ako často bude systém odoberať vzorky z prijatých žiadostí. Parameter je v sekundách. Druhým parametrom bol reqs density per unit, ktorý stanovuje koľko požiadaviek je povolených vo vzorke, ktorú odoberá sampling time unit. Tretím parametrom bol remove latency, ktorý stanovuje ako dlho bude IP adresa držaná v pamäti po poslednej prijatej požiadavke z danej IP adresy.

Po otestovaní som zhodnotil, ktoré zabezpečenie z nich je najvhodnejšie pre systém, ktorý má spĺňať podmienky zabezpečenia a ochrany pred útokmi. Tabuľka 1 obsahuje všetky scenáre, s tým, že posledný trinásty scenár je už testovanie na základe vyhodnotenia správnej ochrannej techniky. Preto nazvem prvých dvanásť scenárov ako parametrizačné scenáre a posledný ako záťažový scenár.

U každého testu som zaznamenával do tabuliek dva typy údajov. Prvá skupina stanovuje ako bude prebiehať daný test, t.j. koľko bude vykonaných hovorov, maximálny počet súbežných hovorov a intenzitu hovorov. Druhou skupinou sú výsledky z testov, kde je uvedený čas trvania testu, počet úspešných a neúspešných telefonátov, a aký bol pomer telefonátov za sekundu. Keďže tabuľky sú veľmi obsiahle, sú obsahom prílohy. Výsledky z parametrizačných testov sú obsahom v Prílohe S strana XV Tabuľka 2 a výsledky zo záťažových testov sú obsahom Prílohy T na strane XVI, Tabuľka 3.

| Číslo testovacieho scenára | Časová vzorka | Počet požiadaviek vo vzorke | Podržanie v pamäti |
|---|---|---|---|
| **1** | 2 | 20 | 4 |
| **2** | 2 | 20 | 24 |
| **3** | 2 | 20 | 14 |
| **4** | 10 | 200 | 4 |
| **5** | 6 | 80 | 4 |
| **6** | 2 | 10 | 4 |
| **7** | 12 | 40 | 4 |
| **8** | 2 | 10 | 12 |
| **9** | 12 | 40 | 20 |
| **10** | 20 | 80 | 4 |
| **11** | 20 | 80 | 20 |
| **12** | 12 | 80 | 20 |
| **13** | 2 | 20 | 4 |

*Tabuľka 1 - Parametrizácia testovacích scenárov*

## 5.3. Zhodnotenie testovania

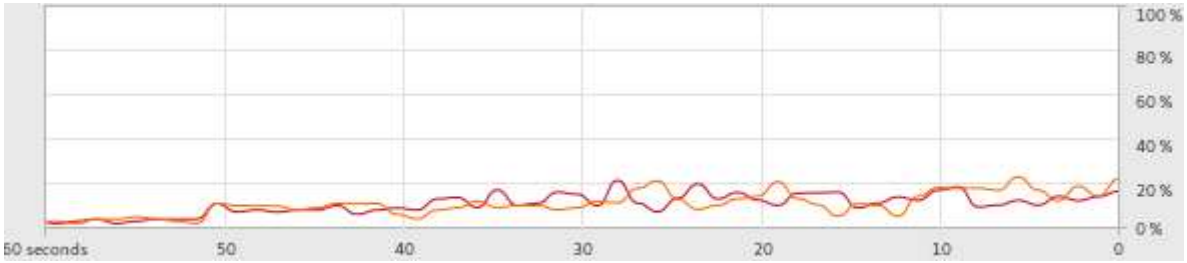
Z testov je vyplýva, že najväčší efekt na blokovanie mal parameter veľkosti okna a počtu vzoriek v okne. Až u scenára číslo deväť je vidieť aj efekt odstránenia zapísanej hodnoty v pamäti keď pri teste č. 109.3 bol znížený počet hovorov za sekundu na dva a už sa zmenil výsledok prijatých telefonátov ku zamietnutým na tridsaťtri ku trom. V teste č. 109.4, tak ako v predošlých testoch č. 106.7, 106.11 či 108.11 som dokázal vytvoriť test kedy všetky hovory boli prijaté a veľkým počtom obslúžených telefonátov. Test č. 108.11 má až štyristo obslúžených telefonátov v priebehu dvesto sekúnd. V danom scenári je veľmi malé okno, o hodnote dva, preto bolo potrebné zmenšiť aj intenzitu hovorov pri teste. U testu č. 110.3 pozorujem veľmi rýchly priebeh testu, kedy je za sekundu vytvorených až deväťdesiatsedem telefonátov. Dosiahnuté je to veľkým parametrom okna, počtu žiadostí v okne a nízkym počtom celkových telefonátov. Jednoducho tie telefonáty sa zmestili do daného okna. Taktiež je možné pozrieť zaťaženie systému na obrázku.

Testy uvedené v Tabuľka 1, obsahujú len testy s platnými registračnými parametrami ako je užívateľské meno, heslo, číslo ktoré je obsahom databáze a taktiež správy mali celistvý formát a neboli ničím porušené. Akonáhle bola posielaná zlá správa, napr. INVOTE namiesto INVITE, po treťom prijatí danej požiadavky v rámci bezpečnostného okna, bola daná IP adresa zablokovaná. To isté platilo ak sa niekto snažil zavolať na neplatné číslo alebo sa prihlásiť so zlým prihlasovacím menom alebo heslom.

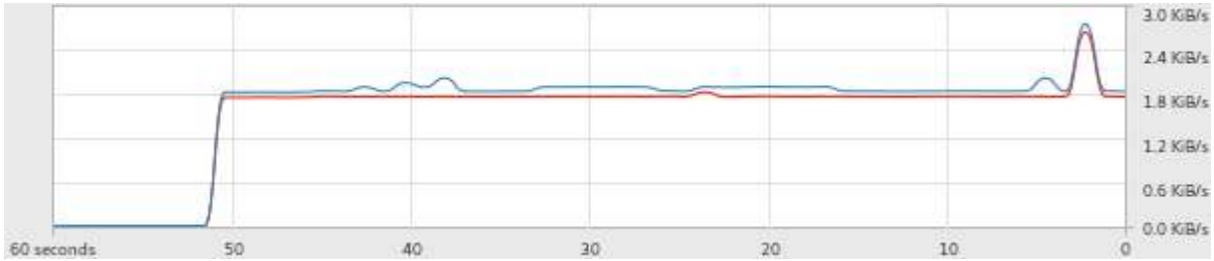
### 5.3.1. Výstupy zo zaťaženia SBC

Výsledky zo všetkých testov nie je možné obsiahnuť v tejto práci, preto uvediem len niektoré zaťaženia procesorov a sieťových prostriedkov. Budú nimi testy 108.11, 42, uvediem rozdiel medzi testom 110.3 a 110.7.

V teste č. 108.11, kde boli všetky telefonáty úspešné a je možné pozorovať len malý náraz na zaťažení systéme.

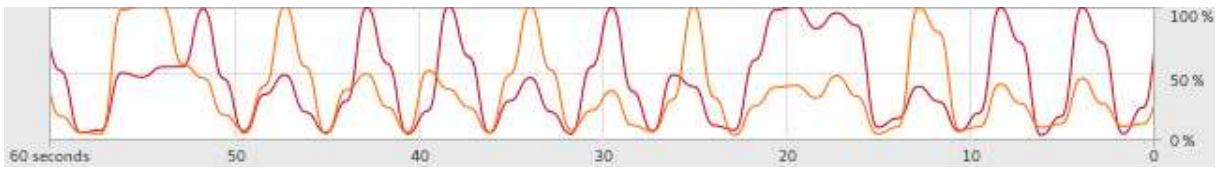



*Ilustrácia 11 - Vyťaženie procesora pri teste č. 108.11*




*Ilustrácia 12- Vyťaženie sieťových prostriedkov pri teste č. 108.11*

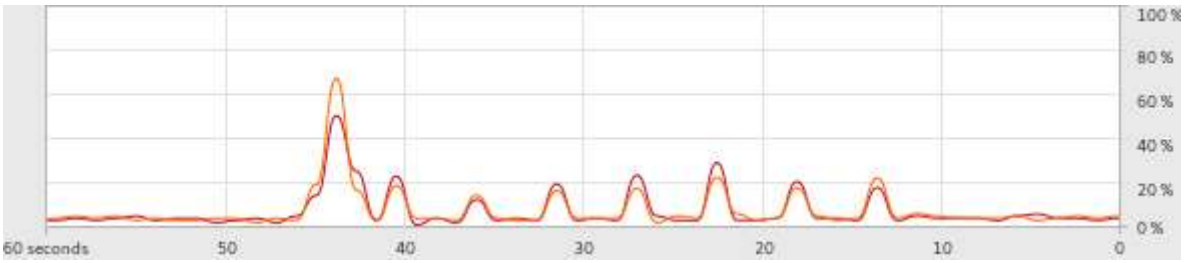
V zaťažovacom teste č. 42 bolo odoslaných veľa žiadostí na server a je možné odčítať z tabuľky, že dva telefonáty boli nakoniec úspešné. Je to vďaka dĺžke zablokovania IP adresy a dĺžke celého testu. Objavila sa tam medzera kedy prekĺzli dva telefonáty, čo je v celkovom počte štyristo tisíc veľmi dobrý výsledok.



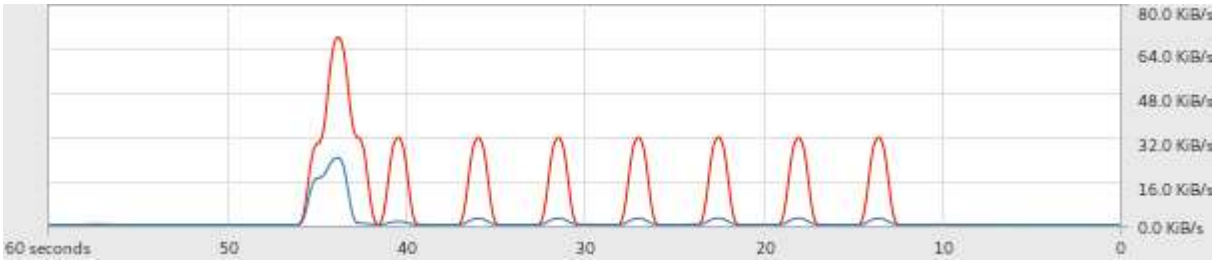

*Ilustrácia 13- Vyťaženie procesora pri teste č. 42*


*Ilustrácia 14- Vyťaženie sieťových prostriedkov pri teste č. 42*

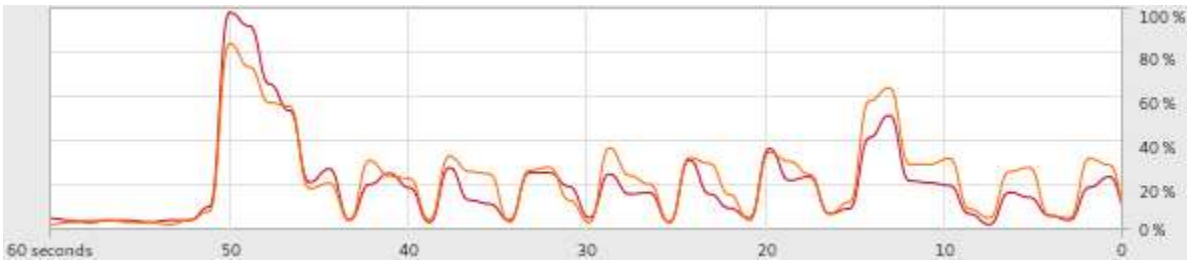
Rozdiel medzi testami 110.3 a 110.7 je v počte vykonaných telefonátov. Z metriky nastavenia zabezpečenia je možné vyčítať, že len štyridsať telefonátov v krátkom časovom úseku môže byť prijatých bez zablokovania. Je to aj dané tým, že testy majú rovnako vysokú intenzitu hovorov za sekundu.



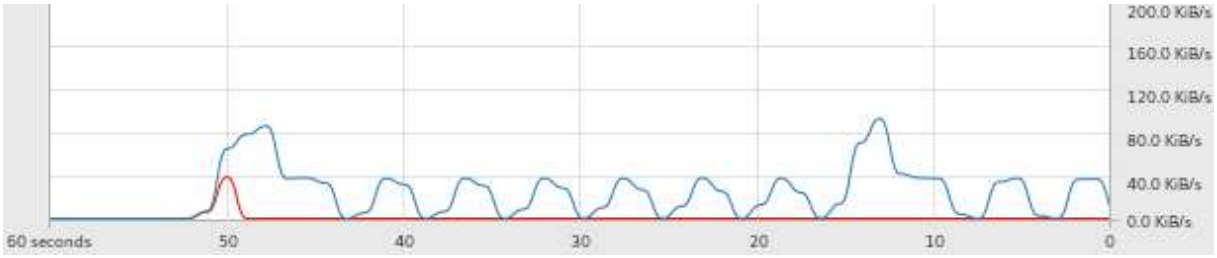

*Ilustrácia 15- Vyťaženie procesora pri teste č. 110.3*




*Ilustrácia 16- Vyťaženie sieťových prostriedkov pri teste č. 110.3*




*Ilustrácia 17- Vyťaženie procesora pri teste č. 110.7*




*Ilustrácia 18- Vyťaženie sieťových prostriedkov pri teste č. 110.7*

# 6 Záver

Výsledkom tejto práce je spojenie dvoch open-source telefónnych systémov za účelom implementácie aspoň časti funkcií SBC, ktoré sú definované v RFC 5853. Implementované a otestované funkcie boli, topology hiding, flood protection, brutal force attack protection, distributed denial of service attack protection a taktiež bola implementovaná zabezpečená komunikácia TLS, ktorá ale nebola, využitá pri výkonnostnom testovaní. Nástrojom na testovanie bol program SIPp. Boli testované rôzne scenáre zabezpečenia skrz modul PIKE. Nakoniec bol vytvorený jeden scenár, kde sa využili všetky zabezpečovacie vyššie spomenuté funkcie a ten bol podrobený 42 útokom s rôznymi scenármi posielania žiadostí. Scenáre útokov sa menili podľa počtu poslaných žiadostí, ich intenzity za jednotku času a počtu súbežných telefonátov. Zistením pri tomto testovaní bolo, že aj keď je systém zaťažený veľkým počtom žiadostí INVITE viz. kapitola 5.2, je neustále schopný registrovať platných užívateľov, presmerovať ich žiadosti a vytvoriť funkčné spojenie medzi nimi. Finálne nastavenie systému s metrikami spomenutými v kapitole 4, bolo nastavené tak, aby reagovalo promptne a dokázalo zachytiť aj rýchle zmeny v útokoch. Také riešenie si, ale zobralo veľkú daň na zaťažení servera, kedy musí neustále spracovávať vzorky a zapisovať ich do tabuľky. Keďže bolo nastavené zapisovanie do logu s vysokou prioritou, zapisovali sa aj tie nepatrné detaily, ktoré tiež odoberali z výkonu daného systému. Po implementácii TLS bolo otestované zabezpečenie, keď sa klient s platným certifikátom dokázal registrovať na server. Problémom pri použití TLS je, že funguje len na systéme, ktorý sám nakonfigurujem a poznám jeho funkčnosť. Ak sa dostanem za hranice svojej siete, nie je isté či aj nasledovné brány a hraničné zariadenia susedných sietí taktiež využívajú zabezpečenie TLS, aj keď to schéma SIPS vyžaduje. Stačí, že niekde v danom komunikačnom reťazi je jeden článok, ktorý nevyužíva vyššiu formu zabezpečenia a daná komunikácia môže byť kompromitovaná.

Prínosom tejto práce bolo zhodnotenie danej problematiky a vytvorenie systému, v ktorom sú implementované časti funkcií SBC, z voľných produktov na trhu, ktoré sú distribuované pod licenciou GPL. Pri scenároch s väčšou voľnosťou nastavenia som zistil, že moje SBC dokáže obsluhovať vysoký počet telefonátov, čo z daného riešenia robí veľmi atraktívny produkt.

# Zoznam použitej literatúry

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, „The Internet Engineering Task Force," June 2002. [Online]. Available: https://www.ietf.org/rfc/rfc3261.txt.

[2] F. E.Goncalves, Building Telephony Systems with OpenSIPS 1.6, Birmingham: Packt Publishing, 2010.

[3] „Index of /pub/kamailio/latest/src," The SIP Router Project, 05 04 2017. [Online]. Available: https://www.kamailio.org/pub/kamailio/latest/src/.

[4] D.-C. Mierla, „A Piece of History: Nine Years SER," The SIP Router Project, 3 Sep 2010. [Online]. Available: http://sip-router.org/wiki/history/ser-9-years.

[5] „Welcome To Kamailio – The Open Source SIP Server," The SIP Router Project, 2008. [Online]. Available: https://www.kamailio.org/w/.

[6] „Kamailio Modules - v5.1.x (devel)," The SIP Router Project, 05 04 2017. [Online]. Available: https://www.kamailio.org/docs/modules/devel/.

[7] L. M. J. V. M. R. Bryant, Asterisk: The Definitive Guide, Sebastopol: O'Reilly, May, 2013.

[8] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, M. Bhatia, „Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments," Internet Engineering Task Force, April 2010. [Online]. Available: https://tools.ietf.org/html/rfc5853.

[9] „Using ODBC in the core," 18 Jan 2013. [Online]. Available: https://wiki.freeswitch.org/wiki/Using_ODBC_in_the_core.

[10] „How to Install the MySQL ODBC Driver on Ubuntu 16.04?," [Online]. Available: https://www.datasunrise.com/blog/how-to-install-the-mysql-odbc-driver-on-ubuntu-16-04/.

[11] „Upgrade to Ubuntu 16.04 libmyodbc missing," Jun 2016. [Online]. Available: https://community.asterisk.org/t/upgrade-to-ubuntu-16-04-libmyodbc-missing/66928.

[12] A. Nagy, „Installing FreePBX 13 on Ubuntu Server 14.04.2 LTS," 02 Feb 2017. [Online]. Available: https://wiki.freepbx.org/display/FOP/Installing+FreePBX+13+on+Ubuntu+Server+14.04.2+LTS.

[13] D.-C. Mierla, „Realtime Integration Of Asterisk 1.4 With Kamailio 1.5.x," 14 Feb 2017. [Online]. Available: https://www.voip-info.org/wiki/view/Realtime+Integration+Of+Asterisk+1.4+With+Kamailio+1.5.x.

[14] A. Kairiukstis, „ODBC.ini for Asterisk PBX res_odbc, cdr_odbc and realtime integration," Nov 2016. [Online]. Available: https://gist.github.com/andrius/d53861ae5929657d0446/.

[15] F. Siquijor, „Asterisk v11.7 Realtime Integration with Kamailio v4.1.X on Ubuntu," 17 Jan 2015. [Online]. Available: http://lextertech.blogspot.sk/2015/01/asterisk-v117-realtime-integration-with.html.

[16] D.-C. Mierla, „Kamailio 4.0.x and Asterisk 11.3.0 Realtime Integration using Asterisk Database," 11 Mar 2013. [Online]. Available: http://kb.asipto.com/asterisk:realtime:kamailio-4.0.x-asterisk-11.3.0-astdb.

[17] „unable to load sip.conf (or iax )," 2015. [Online]. Available: http://asterisk-users.digium.narkive.com/6hPydMSv/11-13-1-unable-to-load-sip-conf-or-iax.

[18] „No such command 'sip show peers' when using asterisk," Feb 2016. [Online]. Available: https://askubuntu.com/questions/426554/no-such-command-sip-show-peers-when-using-asterisk.

[19] Palo, „Kamctl moni - FIFO error problem," 18 Nov 2010. [Online]. Available: http://nil.uniza.sk/sip/kamailio/kamctl-moni-fifo-error-problem.

[20] F. Silva, „Error opening Kamailio's FIFO," 11 Jul 2016. [Online]. Available: https://github.com/sipcapture/homer/issues/179.

[21] K. Mullen, „Error opening Kamailio's FIFO," 06 Dec 2010. [Online]. Available: http://sip-router.1086192.n5.nabble.com/Error-opening-Kamailio-s-FIFO-td22791.html.

[22] D.-C. Mierla, „Kamailio (OpenSER) - Debug and syslog messages," 12 Feb 2015. [Online]. Available: https://www.kamailio.org/dokuwiki/doku.php/tutorials:debug-syslog-messages.

[23] D. W. Graham, „[SR-Users] Debugging and syslog," 16 Mar 2013. [Online]. Available: https://lists.kamailio.org//pipermail/sr-users/2013-March/077217.html.

[24] D.-C. Mierla, „topoh Module," The Kamailio SIP Server Project, 01 Mar 2016. [Online]. Available: http://kamailio.org/docs/modules/4.3.x/modules/topoh.html.

[25] B.-A. Iancu, „pike Module," The SIP Router Project, 01 Mar 2016. [Online]. Available: http://www.kamailio.org/docs/modules/4.3.x/modules/pike.html.

[26] E.-R. Modroiu, „HTable Module," The Kamailio SIP Server Project, 01 Mar 2016. [Online]. Available: http://www.kamailio.org/docs/modules/4.3.x/modules/htable.html.

[27] D. V. D. Moere, „Overview of Security related config snippets,“ 24 Mar 2014. [Online]. Available: https://2600hz.atlassian.net/wiki/display/docs/Kamailio+security+tips.

[28] M. Baptista, „[SR-Users] Kamailio 1.5.4 - Pike module ignoring remove_latancy parameter,“ 10 Sep 2010. [Online]. Available: https://lists.kamailio.org//pipermail/sr-users/2010-September/065337.html.

[29] H. Westerholt, „Pike module,“ 29 Oct 2008. [Online]. Available: http://marc.info/?l=openser-devel&m=122658791609178&w=2.

[30] D.-C. Mierla, „Kamailio 1.5.4 - Pike module,“ 23 Sep 2010. [Online]. Available: https://lists.kamailio.org//pipermail/sr-users/2010-September/065480.html.

[31] D.-C. Mierla, „Experiences from 18 Hours of SIP Scanning Attack,“ ASIPTO, 17 Nov 2010. [Online]. Available: http://kb.asipto.com/kamailio:usage:k31-sip-scanning-attack.

[32] Ovidiu Sas, Bogdan Vasile Harjoc, Hendrik Scholz, „ratelimit Module,“ The Kamailio SIP Server Project, 01 Mar 2016. [Online]. Available: http://www.kamailio.org/docs/modules/4.3.x/modules/ratelimit.html.

[33] O. Sas, „Ratelimit outgoing SIP INVITE,“ 19 Aug 2008. [Online]. Available: https://lists.kamailio.org/pipermail/users/2008-August/019214.html.

[34] O. Sas, „rate limit module,“ 2 Dec 2011. [Online]. Available: https://lists.kamailio.org//pipermail/sr-users/2011-December/071133.html.

[35] D. Escartin, „doubt using rate limit,“ 25 Mar 2016. [Online]. Available: http://sip-router.1086192.n5.nabble.com/doubt-using-rate-limit-td148776.html.

[36] A. Pelinescu-Onciul, „TLS Module,“ The Kamailio SIP Server Project, 01 Mar 2016. [Online]. Available: http://kamailio.org/docs/modules/4.3.x/modules/tls.html.

[37] „Create Certificates to be used with Kamailio,“ OpenSER, 30 Sep 2010. [Online]. Available: http://www.kamailio.org/dokuwiki/doku.php/tls:create-certificates.

[38] Palo, „Configuring TLS support in Kamailio 3.1,“ NIL, 29 Nov 2010. [Online]. Available: http://nil.uniza.sk/network-security/tls/configuring-tls-support-kamailio-31-howto.

[39] „Installing a root/CA Certificate,“ Ubuntu, 12 Jan 2016. [Online]. Available: https://askubuntu.com/questions/73287/how-do-i-install-a-root-certificate.

[40] R. Bryant, L. Madsen, J. V. Meggelen, Asterisk - The Definitive Guide 4th Edition, Sebastopol, CA: O'Reilly.

[41] O. JACQUES, „Welcome to SIPp,“ 20 April 2014. [Online]. Available: http://sipp.sourceforge.net/.

[42] „SIPp 3.5.1,“ 23 Mar 2016. [Online]. Available: https://github.com/SIPp/sipp/releases.

[43] Richard GAYRAUD, Olivier JACQUES, „SIPp reference documentation,“ 20 Mar 2014. [Online]. Available: http://sipp.sourceforge.net/doc/reference.html.

[44] „sipp-scenarios,“ github, 14 May 2011. [Online]. Available: https://github.com/saghul/sipp-scenarios/blob/master/register_data.csv.

[45] „sipp-scenarios,“ github, 17 Mar 2011. [Online]. Available: https://github.com/saghul/sipp-scenarios/blob/master/sipp_uac_register.xml.

# Zoznam príloh

Príloha A: *Vytvorenie virtuálneho stroja v prostredí VMware*

Create a new virtual machine
Configuration: Custom (advanced)
Hardware compatibility: Workstation 12.0
Install from: I will install the operating system later
Guest Operation System: Linux: Ubuntu Linux (64-bit)
Name and Location: server; C:\virtuals
Processors: 1x virtual socket, 2x cores per virtual socket
Memory: 2048 MB
Network connection: Use network address translation (NAT)
SCSI Controller: LSI Logic
Virtual disk type: SCSI
Disk: Create a new virtual disk
Create a Disk: 20GB; Allocate all disk space now; Store with the virtual machine
Virtual machine name and Location: server 1; C:\virtuals
Disk File: server 1.vmdk
Ready to Complete: Finish
Edit Settings -> CD/DVD drive -> Use ISO image file: ubuntu-16.04.1-server-amd64.iso and Enable Connect at power on

Príloha B: *Inštalácia operačného systému Ubuntu Server*

Power on the virtual machine
Language: English
Install Ubuntu Server
Select a language: English - English
Select your location: Other -> Europe -> Slovakia
Configure locales: United States - en_US.UTF-8
Configure the keyboard: No (detecting keyboard) -> Slovak -> Slovak - Slovak (qwerty)
Configure network
Hostname: server
User/Password: student/student
Encrypt of home directory: No
Configure the clock: Yes (timezone Europe/Bratislava)
Partition disks: Guided - use entire disk and set up LVM -> SCSI33 -> Yes (write the changes to disks and configure LVM) -> 21.0GB -> Yes
Yes (write changes to disks)
Configure the package manager:  nič (proxy)
Configuring tasksel: No automatic updates

```
     Software selection: LAMP server, standard system utilities,
OpenSSH server
     Configuring mysql-server-5.7: root/student
     Install the GRUB boot loader to the master boot record: Yes
```

Príloha C: *Konfiguračné súbory pre ODBC konektor*

```
     # nano /etc/odbcinst.ini

[MySQL]
Driver=/usr/lib/x86_64-linux-gnu/odbc/libmyodbc5w.so
UsageCount=2

     # nano /etc/odbc.ini

[MySQL-asterisk]
Description = MySQL Asterisk database
Trace = Off
TraceFile = stderr
Driver = MySQL
SERVER = localhost
USER = asterisk
PASSWORD = asterisk_password
PORT = 3306
DATABASE = asterisk
Socket=/var/run/mysqld/mysqld.sock
option=3
```

Príloha D: *Konfiguračné súbory pre pobočkovú ústredňu Asterisk*

```
     # nano /etc/asterisk/res_odbc.conf

[asterisk]
enabled => yes
pre-connect => yes
sanitysql => select 1
idlecheck => 300
limit => 5
share_connections => yes
isolation => repeatable_read
forcecommit => yes
dsn => MySQL-asterisk
```

```
database => asterisk
username => asterisk
password => asterisk_password
```

```
     # nano /etc/asterisk/extconfig.conf
```

```
[settings]
sipusers => odbc,asterisk,sipusers
sippeers => odbc,asterisk,sipusers
sipregs => odbc,asterisk,sipregs
voicemail => odbc,asterisk,voicemail
```

```
     # nano /etc/asterisk/sip.conf
```

```
[general]
bindaddr=0.0.0.0
transport=udp
bindport=5080
rtcachefriends=yes
```

```
     # nano /etc/asterisk/extensions.conf
```

```
[LocalSets]
exten => _1XX,1,Dial(SIP/${EXTEN})
exten => _1XX,n,Voicemail(${EXTEN},u)
exten => _1XX,n,Hangup
exten => _1XX,101,Voicemail(${EXTEN},b)
exten => _1XX,102,Hangup
```

```
     # nano /etc/asterisk/modules.conf
```

```
[modules]
autoload=yes
;
;odbc
preload => res_odbc.so
;
;gtx console, kde console
noload => pbx_gtkconsole.so
noload => pbx_kdeconsole.so
;
;intercom
noload => app_intercom.so
;
;modem channel driver
```

```
noload => chan_modem.so
noload => chan_modem_aopen.so
noload => chan_modem_bestdata.so
noload => chan_modem_i4l.so
;
;CAI middleware and hardware
noload => chan_capi.so
;
load => res_musiconhold.so
;
;OSS or ALSA load, not both
noload => chan_alsa.so
;noload => chan_oss.so
;
;CDR loggibg to sql lite
noload => cdr_sqlite.so
;
noload => app_directory_odbc.so
;
;asterisk in a database
preload => res_config_odbc.so
noload => res_config_pgsql.so
;
noload => res_pjsip.so
load => chan_sip.so
[global]
```

Príloha E: *Pridanie užívateľov do tabuliek sipusers, sipregs a voicemail*

```
     # INSERT INTO sipusers (name, defaultuser, host, sippasswd,
fromuser, fromdomain, mailbox) VALUES ('101', '101', 'dynamic',
'101', '101', '192.168.30.128', '101');
     # INSERT INTO sipusers (name, defaultuser, host, sippasswd,
fromuser, fromdomain, mailbox) VALUES ('102', '102', 'dynamic',
'102', '102', '192.168.30.128', '102');
     # INSERT INTO sipusers (name, defaultuser, host, sippasswd,
fromuser, fromdomain, mailbox) VALUES ('103', '103', 'dynamic',
'103', '103', '192.168.30.128', '103');

     # INSERT INTO sipregs(name) VALUES('101');
     # INSERT INTO sipregs(name) VALUES('102');
     # INSERT INTO sipregs(name) VALUES('103');
```

```
     # INSERT INTO voicemail(context, mailbox, password) VALUES
('default', '101', '1234');
     # INSERT INTO voicemail(context, mailbox, password) VALUES
('default', '102', '1234');
     # INSERT INTO voicemail(context, mailbox, password) VALUES
('default', '103', '1234');
```

Príloha F: *Definovanie prístupových užívateľov a hesiel pre SIP proxy Kamailio*

```
     # nano /etc/kamailio/kamctlrc

SIP_DOMAIN=192.168.30.128
DBENGINE=MYSQL
DBHOST=localhost
DBNAME=kamailio
DBRWUSER="kamailio"
DBRWPW="kamailiorw"
DBROUSER="kamailioro"
DBROPW="kamailioro"
ALIASES_TYPE="DB"
CTLENGINE="FIFO"
OSER_FIFO="/tmp/kamailio_fifo"
VERBOSE=1
PID_FILE=/var/run/kamailio.pid
```

Príloha G: *Kolokované spojenie SIP proxy Kamailio a pobočkovej ústredne Asterisk v konfiguračnom súbore kamailio.cfg*

```
     # nano /etc/kamailio/kamailio.cfg

#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
#!define WITH_ASTERISK

#!define DBURL "mysql://kamailio:kamailiorw@localhost/kamailio"
#!ifdef WITH_ASTERISK
#!define DBASTURL
"mysql://asterisk:asterisk_password@localhost/asterisk"

#!ifdef WITH_ASTERISK
asterisk.bindip = "192.168.30.128" desc "Asterisk IP Address"
```

```
asterisk.bindport = "5080" desc "Asterisk Port"
kamailio.bindip = "192.168.30.128" desc "Kamailio IP Address"
kamailio.bindport = "5060" desc "Kamailio Port"
#!endif

mpath="modules_k:modules"
#!else
mpath="/usr/lib/x86_64-linux-
gnu/kamailio/modules/_k/:/usr/lib/x86_64-linux-
gnu/kamailio/modules/"

# ----- mi_fifo params -----
modparam("mi_fifo", "fifo_name", "/tmp/kamailio_fifo")
```

Príloha H: *Výstup z príkazu KAMCTL MONI*

```
[cycle #: 1706; if constant make sure server lives]
Server:: kamailio (4.3.4 (x86_64/linux))
Build:: mi_core.c compiled with gcc 5.3.1
Flags:: STATS: Off, USE_TCP, USE_TLS, USE_SCTP, TLS_HOOKS,
USE_RAW_SOCKS, DISABLE_NAGLE, USE_MCAST, DNS_IP_HACK, SHM_MEM,
SHM_MMAP, PKG_MALLOC, DBG_QM_MALLOC, USE_FUTEX, FAST_LOCK-
ADAPTIVE_WAIT, USE_DNS_CACHE, USE_DNS_FAILOVER, USE_NAPTR,
USE_DST_BLACKLIST, HAVE_RESOLV_RES
GIT:: 0ec860
Now:: Sat Apr 22 13:06:42 2017
Up since:: Sat Apr 22 12:08:42 2017
Up time:: 3480 [sec]

Transaction Statistics:
tmx:UAS_transactions = 167
tmx:UAC_transactions = 102
tmx:inuse_transactions = 1

Stateless Server Statistics:
sl:sent_replies = 213
sl:sent_err_replies = 0
sl:received_ACKs = 2

UsrLoc Stats:
usrloc:location-contacts = 2
usrloc:location-expires = 2
usrloc:location-users = 2
usrloc:registered_users = 2
```

*Ilustrácia 19 - Kamailio monitoring vybavených či prijatých transakcií*

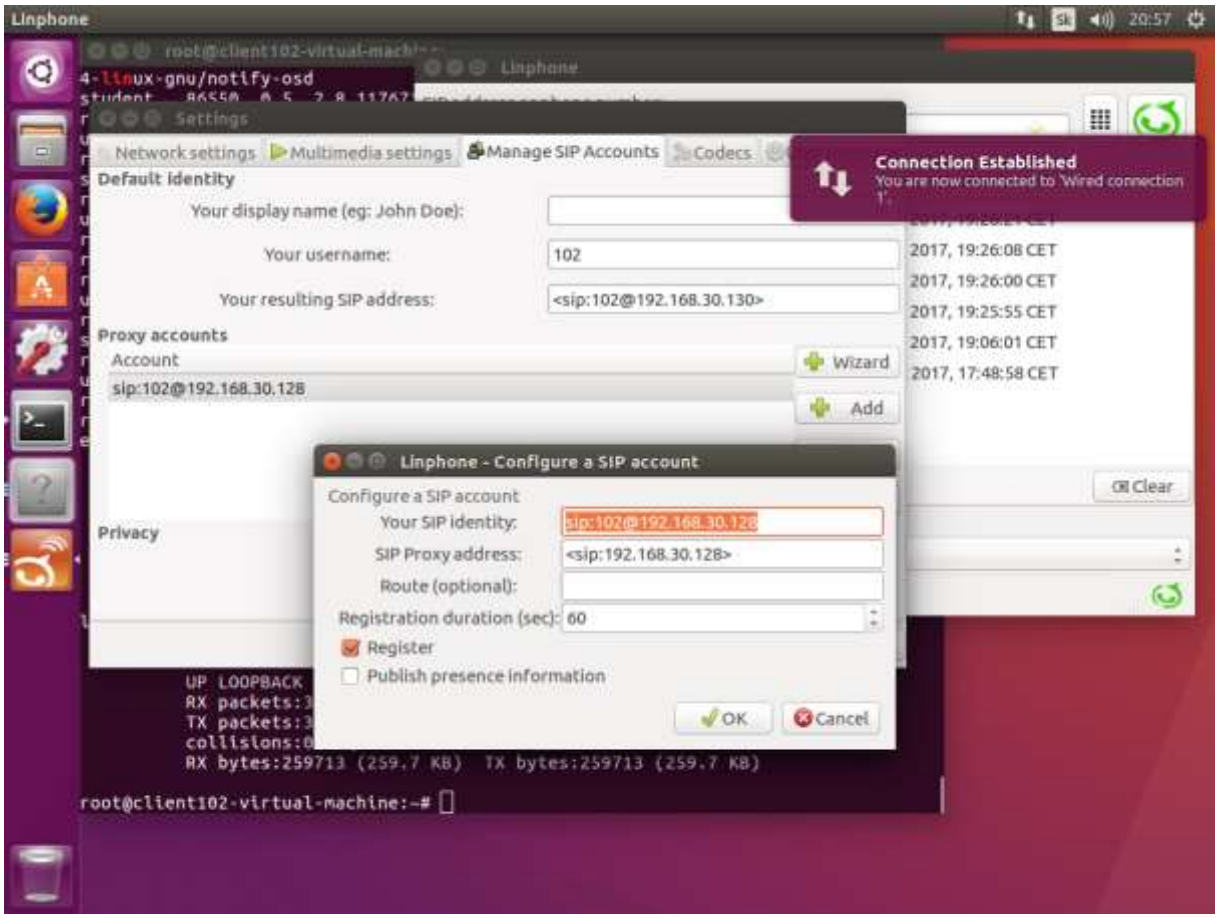
Príloha I: *Inštalácia programu Linphone a nastavenie hovoru*

Linphone sa nachádza v odkazoch na repozitáre a jeho inštalácia bola jednoduchá

```
# apt-get install linphone
```

Nastavenie Linphone pre účely vykonávania hovoru

```
(move to top of the screen) Options-> manage sip accounts -> add
Your username: 102
YOUR SIP identity: sip:102@192.168.30.128
SIP Proxy address: <sip:192.168.30.128>
```




*Ilustrácia 20 - Nastavenie užívateľa v programe Linphone*

Príloha J: *Inštalácia programu Wireshark, odchytenie spojenia a nastavenie filtru*

Inštalácia programu Wireshark cez konzolu

```
# apt-get install wireshark
```

Príkaz na vytvorenie záznamu v konzole

```
# tcpdump -nq -s 0 -i any -w /tmp/dump.pcap
```

Pre ľahšiu prácu a separovanie len potrebnej komunikácie som najčastejšie využil filter

```
udp.port==5060 || tcp.port == 5060 || udp.port==5080 ||
tcp.port==5080
```

Príloha K: *Topology hiding INVITE IN*

```
Session Initiation Protocol (INVITE)
    Request-Line: INVITE sip:103@192.168.30.128 SIP/2.0
    Message Header
        Via: SIP/2.0/UDP
192.168.30.132:5060;rport;branch=z9hG4bK1961259976
        From: <sip:102@192.168.30.128>;tag=218566112
        To: <sip:103@192.168.30.128>
        Call-ID: 1264314768
        CSeq: 21 INVITE
        Contact: <sip:102@192.168.30.132>
        Proxy-Authorization: Digest username="102",
realm="192.168.30.128", nonce="WNe9V1jXvCvoW4Q+LGt0CMgtqst/miYk",
uri="sip:103@192.168.30.128",
response="70cc3c1c6aa09b285cf8f3aece7c7fa0", algorithm=MD5
        Content-Type: application/sdp
        Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY,
MESSAGE, SUBSCRIBE, INFO
        Max-Forwards: 70
        User-Agent: Linphone/3.6.1 (eXosip2/4.1.0)
        Subject: Phone call
        Content-Length:   438
    Message Body
```

Príloha L: *Topology hiding INVITE OUT*

```
Session Initiation Protocol (INVITE)
    Request-Line: INVITE sip:103@192.168.30.128 SIP/2.0
    Message Header
        Record-Route: <sip:192.168.30.128;lr=on;ftag=218566112>
```

```
        Via: SIP/2.0/UDP
192.168.30.128;branch=z9hG4bKdcad.39ba82267e0edf5aeb0e23fa195509b9.0
        Via: SIP/2.0/UDP 10.1.1.10;branch=z9hG4bKsr-
lzWTJDY9FNmSaKkUFHB2JhZsoNjDFNjQFDYd4HksFnq2ci4W6VcWcnzQoHY9FH5jJhFI
JhZDFhqlCnyB6V5tA0E7A0Tm4HksFnqRAwK95sUmphWraDaRCDZM4hZ24HBM4D5*
        From: <sip:102@192.168.30.128>;tag=218566112
        To: <sip:103@192.168.30.128>
        Call-ID: 1264314768
        CSeq: 21 INVITE
        Contact: <sip:10.1.1.10;line=sr-
AsWIohZIFBkQoHY9FH5jJhFIJhZDFU**>
        Content-Type: application/sdp
        Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY,
MESSAGE, SUBSCRIBE, INFO
        Max-Forwards: 69
        User-Agent: Linphone/3.6.1 (eXosip2/4.1.0)
        Subject: Phone call
        Content-Length:   438
    Message Body
```

Príloha M: *Konfiguračný súbor pre TLS.cfg*

```
[server:default]
method = TLSv1
verify_certificate = yes
require_certificate = no
private_key = /etc/kamailio/certs/sip.server.com/key.pem
certificate = /etc/kamailio/certs/sip.server.com/cert.pem
ca_list = /etc/kamailio/certs/demoCA/cert.pem

[server:192.168.30.128:5061]
method = SSLv23
verify_certificate = no
require_certificate = no
private_key = /etc/kamailio/certs/sip.server.com/key.pem
certificate = /etc/kamailio/certs/sip.server.com/cert.pem
```

Príloha N: *Vytvorenie certifikátu a privátneho kľúča*

Pre vytvorenie certifikátov je treba upraviť openssl.conf na viac flexibilnú politiku policy_anything.

```
nano /etc/ssl/openssl.cnf

policy          = policy_anything
```

Pokračujem vytvorením adresára a pridelením práv kde sa CA budú generovať.

```
mkdir /etc/certs
chmod 0700 /etc/certs
cd /etc/certs
```

Na k vytvoreniu certifikátu pre CA vytvorím index súbor, ktorý naplním prvým číslom v postupnosti a vygenerujem CA na dobu 10 rokov.

```
mkdir demoCA
cd demoCA
mkdir newcerts
echo '01' > serial
touch index.txt
openssl req -new -x509 -extensions v3_ca -keyout key.pem -out
cert.pem -days 3650
```

Je možné si overiť vygenerovanie CA certifikátov.

```
openssl x509 -in cert.pem -noout -text
openssl x509 -in cert.pem -noout -dates
openssl x509 -in cert.pem -noout -purpose
cd ..
```

Teraz pristúpim k vytváraniu certifikátov pre prístup na server

```
cd /etc/certs/
mkdir sip.server.com
cd sip.server.com
openssl req -new -nodes -keyout key.pem -out req.pem
cd ..
openssl ca -days 730 -out sip.server.com/cert.pem -keyfile
demoCA/key.pem -cert demoCA/cert.pem -infiles sip.server.com/req.pem
```

Nakopírujem si dané certifikáty o servrra to temp odkiaľ je možné skopírovať na local pomocou winscp

```
cp /etc/certs/sip.server.com/cert.pem /tmp/cert.pem
cp /etc/certs/sip.server.com/key.pem /tmp/key.pem
winscp to local
```

Pristúpim ku klientovi kde mu pomocou winscp nakopírujem certifikát aj kľúč do tempu a odtiaľ pridám dané certifikáty pre klienta

```
winscp to tmp
```

```
cp /tmp/cert.pem /usr/local/share/ca-certificates/cert.pem
cp /tmp/key.pem /usr/local/share/ca-certificates/key.pem
update-ca-certificates
```

Overiť správnosť je možné pomocou prílazu openssl x509

```
openssl x509 -in /usr/local/share/ca-certificates/cert.pem -noout -
text
```

Príloha O: *Potrebné balíčky na inštaláciu SIPp*

```
apt-get install pcaputils
apt-get install libncursesw5-dev
apt-get install libncurses5-dev
apt-get install libpcap-dev
apt-get install openssl
apt-get install libssl-dev
apt-get install libssl1.0.0
apt-get install libssl0.9.8
```

Príloha P: *Konfiguračný súbor register-data.csv*

```
SEQUENTIAL
# user, domain, authentication
101;192.168.30.128;[authentication username=101 password=101]
```

Príloha Q: *Konfiguračný súbor register.xml*

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="registration">

<send retrans="500">
<![CDATA[
REGISTER sip:[field1] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
Max-Forwards: 70
From: "sipp" <sip:[field0]@[field1]>;tag=[call_number]
To: "sipp" <sip:[field0]@[field1]>
```

```
Call-ID: reg///[call_id]
CSeq: 7 REGISTER
Contact: <sip:sipp@[local_ip]:[local_port]>
Expires: 3600
Content-Length: 0
User-Agent: SIPp
]]>
</send>

<recv response="100" optional="true">
</recv>

<recv response="401" auth="true" rtd="true">
</recv>

<send retrans="500">
<![CDATA[
REGISTER sip:[field1] SIP/2.0
Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
Max-Forwards: 70
From: "sipp" <sip:[field0]@[field1]>;tag=[call_number]
To: "sipp" <sip:[field0]@[field1]>
Call-ID: reg///[call_id]
CSeq: 8 REGISTER
Contact: <sip:sipp@[local_ip]:[local_port]>
Expires: 3600
Content-Length: 0
User-Agent: SIPp
[field2]
]]>
</send>

<recv response="100" optional="true">
</recv>

<recv response="200">
</recv>

<ResponseTimeRepartition value="10, 20"/>
<CallLengthRepartition value="10"/>
</scenario>
```

Príloha R: *Výpis z testovacieho scenára*

```
------------------------------ Scenario Screen -------- [1-9]:
Change Screen --
  Call-rate(length)   Port   Total-time  Total-calls  Remote-host
 200.0(0 ms)/1.000s   5060      13.06 s         2611
192.168.30.128:5060(UDP)

  200 new calls during 1.004 s period    1 ms scheduler resolution
  2608 calls (limit 10000)               Peak was 2608 calls, after
13 s
  0 Running, 2609 Paused, 1209 Woken up
  0 dead call msg (discarded)            36 out-of-call msg
(discarded)
  3 open sockets

                                  Messages  Retrans   Timeout
Unexpected-Msg
    REGISTER ---------->          2611      8128      0
         100 <----------          0         0         0         0
         401 <----------  E-RTD1  4         0         0         0
    REGISTER ---------->          4         5         0
         100 <----------          0         0         0         0
         200 <----------          3         0         0         0
------ [+|-|*|/]: Adjust rate ---- [q]: Soft exit ---- [p]: Pause
traffic -----

Last Error: Discarding message which can't be mapped to a known SIPp
cal...
------------------------------ Variables Screen -------- [1-9]:
Change Screen --
Action defined Per Message :
=> No action found on any messages

------------------------------ Test Terminated -----------------

------------------------------ Statistics Screen ------- [1-9]:
Change Screen --
  Start Time             | 2017-04-03   01:08:59.661084
1491174539.661084
  Last Reset Time        | 2017-04-03   01:09:56.228832
1491174596.228832
  Current Time           | 2017-04-03   01:09:56.232497
1491174596.232497
-------------------------+---------------------------+-----------
  Counter Name           | Periodic value            | Cumulative
value
-------------------------+---------------------------+-----------
```

```
  Elapsed Time           | 00:00:00:003000           |
00:00:56:571000
  Call Rate              |    0.000 cps              |   88.385 cps
-------------------------+---------------------------+---------
 Incoming call created  |        0                  |        0
  OutGoing call created  |        0                  |     5000
  Total Call created     |                           |     5000
  Current Call           |        0                  |
-------------------------+---------------------------+----------
  Successful call        |        0                  |        3
  Failed call            |        0                  |     4997
-------------------------+---------------------------+----------
  Response Time 1        | 00:00:00:000000           |
00:00:00:000000
  Call Length            | 00:00:00:000000           |
00:00:31:521000
------------------------------ Test Terminated ----------------

2017-04-03      01:09:56.214389 1491174596.214389: Aborting call on
UDP retransmission timeout for Call-ID '5000-2551@192.168.30.131'.
sipp: There were more errors, see 'register_2551_errors.log' file
```

Príloha S: *Výsledky parametrizačných testov*

| Číslo testu | Počet hovo-rov | Maximá-lny počet súbež-ných hovorov | Inten-zita hovo-rov | Čas testu [s] | Úspešné žiadosti | Neúspešné žiadosti | Telefonáty za jednotku času |
|---|---|---|---|---|---|---|---|
| **101** | 50 | 200 | 4000 | 31.56 | 0 | 50 | 1.584 |
| **102** | 50 | 200 | 4000 | 31.59 | 0 | 50 | 1.582 |
| **103** | 50 | 200 | 4000 | 31.6 | 0 | 50 | 1.582 |
| **104** | 50 | 200 | 4000 | 0.06 | 50 | 0 | 724.638 |
| **104.1** | 4000 | 400 | 4000 | 315.58 | 138 | 3862 | 12.675 |
| **104.2** | 4000 | 1000 | 400000 | 127.35 | 19 | 3981 | 31.408 |
| **104.3** | 4000 | 1000 | 400 | 129.09 | 88 | 3912 | 30.985 |
| **105** | 50 | 200 | 4000 | 0.23 | 50 | 0 | 206.612 |
| **105.1** | 4000 | 400 | 4000 | 327.07 | 27 | 3973 | 12.23 |
| **105.2** | 4000 | 1000 | 400000 | 126.23 | 0 | 4000 | 31.687 |
| **105.3** | 4000 | 1000 | 400 | 128.69 | 47 | 3953 | 31.081 |
| **106** | 50 | 200 | 4000 | 31.55 | 0 | 50 | 1.584 |
| **106.1** | 4000 | 400 | 4000 | 315.6 | 5 | 3995 | 12.674 |
| **106.2** | 4000 | 1000 | 400000 | 126.25 | 0 | 4000 | 31.681 |
| **106.3** | 4000 | 1000 | 400 | 128.68 | 5 | 3995 | 31.084 |
| **106.4** | 50 | 2 | 400 | 527.91 | 18 | 32 | 0.095 |
| **106.5** | 20 | 1 | 1 | 20 | 20 | 0 | 0.999 |
| **106.6** | 20 | 10 | 1 | 20 | 20 | 0 | 1 |
| **106.7** | 200 | 40 | 1 | 200 | 200 | 0 | 1 |
| **106.8** | 40 | 200 | 1 | 40 | 40 | 0 | 1 |
| **106.9** | 40 | 200 | 3 | 44.9 | 10 | 30 | 0.891 |
| **106.10** | 40 | 200 | 2 | 20 | 40 | 0 | 1.999 |
| **106.11** | 200 | 200 | 2 | 100.01 | 200 | 0 | 2 |
| **107** | 40 | 200 | 10 | 35.54 | 25 | 15 | 1.125 |
| **107.1** | 40 | 200 | 6 | 38.21 | 30 | 10 | 1.047 |
| **108** | 40 | 200 | 10 | 35.54 | 8 | 32 | 1.125 |
| **108.4** | 50 | 2 | 400 | 527.77 | 18 | 32 | 0.095 |
| **108.6** | 20 | 10 | 1 | 20.01 | 20 | 0 | 0.999 |
| **108.8** | 40 | 200 | 1 | 40.01 | 40 | 0 | 1 |
| **108.9** | 40 | 200 | 3 | 44.88 | 8 | 32 | 0.891 |
| **108.10** | 40 | 200 | 2 | 20.01 | 40 | 0 | 1.999 |
| **108.11** | 400 | 200 | 2 | 200 | 400 | 0 | 2 |
| **109** | 40 | 200 | 10 | 35.54 | 25 | 15 | 1.125 |
| **109.1** | 40 | 200 | 20 | 33.54 | 25 | 15 | 1.192 |
| **109.2** | 200 | 1000 | 100 | 33.54 | 25 | 175 | 5.961 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **109.3** | 40 | 200 | 2 | 51.54 | 33 | 7 | 0.776 |
| **109.4** | 40 | 200 | 1 | 40 | 40 | 0 | 1 |
| **110** | 40 | 200 | 10 | 4.01 | 40 | 0 | 9.955 |
| **110.1** | 40 | 200 | 20 | 2 | 40 | 0 | 19.92 |
| **110.2** | 40 | 200 | 40 | 1 | 40 | 0 | 39.643 |
| **110.3** | 40 | 200 | 100 | 0.4 | 40 | 0 | 97.561 |
| **110.4** | 200 | 1000 | 100 | 33.54 | 50 | 150 | 5.961 |
| **110.5** | 200 | 800 | 100 | 33.54 | 50 | 150 | 5.962 |
| **110.6** | 200 | 800 | 200 | 32.77 | 49 | 151 | 6.103 |
| **110.7** | 4000 | 200 | 100 | 635.49 | 200 | 3800 | 6.294 |
| **110.8** | 4000 | 200 | 40 | 636.3 | 202 | 3798 | 6.286 |
| **110.9** | 1000 | 200 | 20 | 167.77 | 50 | 950 | 5.96 |
| **110.10** | 400 | 40 | 40 | 266.09 | 100 | 300 | 1.503 |
| **110.12** | 200 | 200 | 4 | 81.54 | 64 | 136 | 2.453 |
| **110.13** | 200 | 200 | 2 | 100.01 | 200 | 0 | 2 |
| **111** | 40 | 200 | 10 | 4.01 | 40 | 0 | 9.958 |
| **111.1** | 40 | 200 | 20 | 2.01 | 40 | 0 | 19.891 |
| **111.2** | 40 | 200 | 40 | 1.01 | 40 | 0 | 39.37 |
| **111.3** | 40 | 200 | 100 | 0.4 | 40 | 0 | 97.8 |
| **111.4** | 200 | 1000 | 100 | 33.54 | 50 | 150 | 5.961 |
| **111.5** | 200 | 800 | 100 | 33.57 | 50 | 150 | 5.957 |
| **111.6** | 200 | 800 | 200 | 32.54 | 50 | 150 | 6.144 |
| **112** | 40 | 200 | 10 | 4 | 40 | 0 | 9.973 |
| **112.1** | 40 | 200 | 20 | 2 | 40 | 0 | 19.91 |
| **112.2** | 200 | 1000 | 100 | 33.54 | 49 | 151 | 5.961 |

*Tabuľka 2 - Výsledky parametrizačných testov*

Príloha T: *Výsledky zo záťažových testov*

| **Číslo testu** | **Počet hovo-rov** | **Maximá-lny počet súbež-ných hovorov** | **Intenzita hovorov** | **Čas testu [s]** | **Úspešné žiadosti** | **Neúspešné žiadosti** | **Telefonáty za jednotku času** |
|---|---|---|---|---|---|---|---|
| **1** | 500 | 10000 | 200 | 34.05 | 3 | 497 | 14.683 |
| **2** | 5000 | 10000 | 200 | 13.06 | 3 | 4997 | 88.385 |
| **3** | 60 | 3 | 200 | 587.36 | 6 | 54 | 0.102 |
| **4** | 60 | 1 | 1 | 1484.54 | 16 | 44 | 0.04 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **5** | 50 | 1 | 200 | 1213.82 | 14 | 36 | 0.041 |
| **6** | 10 | 1 | 10 | 190.04 | 4 | 6 | 0.053 |
| **7** | 14 | 2 | 100 | 189.3 | 3 | 11 | 0.074 |
| **8** | 14 | 1 | 100 | 303.49 | 5 | 9 | 0.046 |
| **9** | 14 | 3 | 100 | 126.22 | 3 | 11 | 0.111 |
| **10** | 14 | 1 | 200 | 303.47 | 5 | 9 | 0.046 |
| **11** | 14 | 1 | 500 | 303.46 | 5 | 9 | 0.046 |
| **12** | 14 | 1 | 1000 | 303.45 | 5 | 9 | 0.046 |
| **13** | 14 | 2 | 200 | 189.26 | 3 | 11 | 0.074 |
| **14** | 14 | 2 | 500 | 189.28 | 3 | 11 | 0.074 |
| **15** | 14 | 2 | 1000 | 189.27 | 3 | 11 | 0.074 |
| **16** | 14 | 1 | 10000 | 303.44 | 5 | 9 | 0.046 |
| **17** | 14 | 2 | 10000 | 189.25 | 3 | 11 | 0.074 |
| **18** | 14 | 3 | 200 | 126.2 | 3 | 11 | 0.111 |
| **19** | 14 | 3 | 500 | 126.17 | 3 | 11 | 0.111 |
| **20** | 14 | 3 | 1000 | 126.17 | 3 | 11 | 0.111 |
| **21** | 14 | 3 | 10000 | 126.18 | 3 | 11 | 0.111 |
| **22** | 20 | 5 | 500 | 126.2 | 3 | 17 | 0.158 |
| **23** | 50 | 200 | 10000 | 31.55 | 0 | 50 | 1.584 |
| **24** | 50 | 200 | 20000 | 31.55 | 0 | 50 | 1.584 |
| **25** | 50 | 200 | 50000 | 31.55 | 0 | 50 | 1.584 |
| **26** | 50 | 200 | 100000 | 31.55 | 0 | 50 | 1.584 |
| **27** | 50 | 400 | 10000 | 31.55 | 0 | 50 | 1.584 |
| **28** | 50 | 600 | 10000 | 31.55 | 0 | 50 | 1.584 |
| **29** | 50 | 800 | 10000 | 31.55 | 0 | 50 | 1.584 |
| **30** | 50 | 1000 | 10000 | 31.55 | 0 | 50 | 1.584 |
| **31** | 50 | 2000 | 10000 | 31.55 | 0 | 50 | 1.584 |
| **32** | 200 | 1000 | 10000 | 31.56 | 0 | 200 | 6.336 |
| **33** | 400 | 2000 | 10000 | 31.58 | 0 | 400 | 12.662 |
| **34** | 1000 | 4000 | 50000 | 31.59 | 0 | 1000 | 31.65 |
| **35** | 200 | 4000 | 10000 | 31.56 | 0 | 200 | 6.335 |
| **36** | 400 | 4000 | 10000 | 31.58 | 0 | 400 | 12.662 |
| **37** | 1000 | 4000 | 10000 | 31.65 | 0 | 1000 | 31.59 |
| **38** | 4000 | 20000 | 200000 | 31.7 | 0 | 4000 | 126.179 |
| **39** | 10000 | 20000 | 50000 | 31.91 | 0 | 10000 | 313.234 |
| **40** | 4000 | 10000 | 500000 | 31.67 | 0 | 4000 | 126.259 |
| **41** | 20000 | 2000000 | 5000000 | 32.65 | 0 | 20000 | 612.445 |
| **42** | 400000 | 20000 | 5000000 | 647.47 | 2 | 399998 | 617.785 |

*Tabuľka 3 - Výsledky zo záťažových testov*